Information Presentation in Spoken Dialogue Systems

Vera Demberg

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh

2005

Abstract

To tackle the problem of presenting a large number of alternative options in spoken dialogue systems, two different strategies were used in recent work. Moore *et al.* [2004] exploit information from a user model to select a small set of relevant options based on decision theory. Polifroni *et al.* [2003] use a clustering algorithm to summarize options and allow for stepwise query refinement in order to narrow down to an interesting option.

In this thesis, I show how the benefits of the two approaches can be combined in an approach that identifies compelling options based on a user model and presents complex tradeoffs between alternative options explicitly to allow the user to make a more informed choice.

When there are multiple attractive options, the information is split up into several turns and structured such that the user can gradually refine her request to find the optimal tradeoff. The effectiveness of the dialogue structure is thereby optimized by taking into account the user's valuation.

I demonstrate how the approach can be implemented, and integrated into an existing dialogue system.

Finally, I report the results of an evaluation involving 38 subjects, which shows that my system developed presents complex tradeoffs understandably, increases overall user satisfaction, and significantly improves the user's overview of the available options. Furthermore, my results suggest that presenting users with a brief summary of the irrelevant entities increases user's confidence that they have heard about all relevant options.

Acknowledgements

First of all, I'd like to thank my supervisor Johanna Moore for the fruitful discussions and helpful advice. I want to express my appreciation to Steve Conway for helping me with many technical issues on the FLIGHTS system, to Michael White for adapting the realizer and for his patient answers to my questions on the sentence planner. I also want to thank Neide Franca Rocha who introduced me to E-Prime and shared her experience on experiment design with me. Many thanks to Frank Eisele and my brother, Jens, for their useful comments on my thesis.

I'm also grateful to Evangelisches Studienwerk e.V. Villigst for funding my MSc course.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Vera Demberg*)

Table of Contents

List of Figures

Chapter 1

Introduction

The goal of *spoken dialogue systems* is to offer an efficient and natural way of accessing information services of various types that range from product recommendation, to access to emails, to personal banking. The use of systems with speech interaction is particularly attractive for applications where the user communicates via telephone or cannot use her hands and eyes to operate the system. A challenging issue in spoken dialogue systems, which this work addresses, is the restriction that (possibly complex) information has to be presented linearly.

In traditional systems, all different options and their respective attributes are simply enumerated. But if a lot of complex information is to be conveyed, this form of presentation makes it difficult for the user to remember the various aspects of multiple options and to compare them in memory, especially if not all of the user's concentration can be focused on the task (for example in dialogue systems that are used while driving a car).

The recent evaluation of 9 dialogue systems, Walker *et al.* [2001] has shown that *shorter task duration* correlates with higher user satisfaction. Possible ways to shorten dialogue duration are to identify a small subset of relevant options and present only these in detail or to allow for stepwise refinement.
An approach to identify particularly promising options has first been applied to spoken dialogue systems by Walker *et al.* [2002], who applied user model based multi-attribute decision theory as proposed in Carenini and Moore [2001], to determine which options are particularly relevant to a user. Evaluation showed (see Walker *et al.* [2004]),

that tailoring recommendations and option comparisons to the user increases argument effectiveness and leads to higher user satisfaction with the system. The texts generated by that first system were not very sophisticated. It enumerated options and their attributes which had been determined in the user model based selection process.
Moore *et al.* [2004] tackled the tasks of generating user-tailored descriptions and of presenting options in an *easily memorable and understandable* way. Coherent and naturalness of descriptions were increased by supporting the informational structure with an appropriate intonation, relating options as well as their attributes to one another through the use of discourse cues and scalar terms and introducing referencing expressions.
A possible shortcoming of mentioning relevant options only is that the information space is not fully accessible to the user. *User confidence* in the system might be decreased, because the user feels that the system is hiding information from her, especially if she has no possibility to override her user model.
A different approach to shortening dialogue duration, which does not take into account a user model, was proposed by Polifroni *et al.* [2003]. The aim was to support the user in the query refinement process by generating summarizations of the available flights. These summarizations were based on attribute clusters which were sensitive to the data subset relevant in the current dialogue context. The clustering also increased flexibility for coping with on-line content. The dialogue flow in the Polifroni system is mainly controlled by the user, who can at any stage request any refinement.

In this thesis, I show how the benefits of these recent approaches can be combined to a system that integrates user modelling with automated clustering. In my approach, relevant options are selected based on a user model. A cluster-based tree structure orders them to allow for stepwise refinement when there are multiple relevant options. The effectiveness of the tree structure, which directs the dialogue flow, is optimized by taking into account the user's valuation. Complex tradeoffs between alternative options are presented explicitly to allow for a better overview and a more informed choice.
The achievements of the FLIGHTS system, namely control of intonation and use of discourse cues, scalar terms and referencing expressions are adapted in my system.
I address the issue of maintaining user confidence despite selecting only the relevant options by briefly accounting for the remaining options.

The evaluation of my system with respect to the system presented in Polifroni *et al.* [2003] assessed overall user satisfaction as well as further criteria such as understandability, goodness of overview provided, confidence in having heard about all relevant options, and how quickly the optimal option could be accessed.
38 participants rated five dialogues from each system with respect to these aspects. The results suggest that user model-based clustering leads to higher user satisfaction. Presenting alternative tradeoffs explicitly, significantly improves the user's overview of the available options. Structuring options according to the user's valuations gives users the impression that they could access the relevant options more quickly. Accounting for options that were not presented in detail proved to increase the user's confidence in having heard all relevant options. Although system turns were on average longer, subjects did not report impaired understandability.

While the principal goal of this MSc project was to develop domain independent techniques for presenting options that meet user constraints in spoken dialogue systems, the approach to the problem has been implemented within FLIGHTS, an end-to-end spoken dialogue system for flight booking, recently built at the University of Edinburgh.
As a synthesis of two recently developed approaches, my work represents a timely contribution to the research area of spoken dialogue system design.

Chapter 2 of this thesis begins with an outline of past work in dialogue systems. I then explain the conceptual design which underlies my system (chapter 3). In chapter 4, I give an overview of the system architecture and go through the steps which I undertook in the course of this project.
Chapter 5 presents the algorithms I used for the implementation of my ideas and addresses main design issues. The integration of my changes into the FLIGHTS system and adaptation of its components is explained in chapter 6.
The system's performance with respect to user satisfaction, understandability, accessibility of options and goodness of overview were assessed with user-evaluation. The design of the user evaluation is presented in chapter 7. Chapter 8 analyses and discusses the results.
Finally, I draw conclusions and suggest further work in chapter 9.

Chapter 2

Background

2.1 Traditional Dialogue Systems

Traditional spoken dialogue systems usually present a set of options to the user that match her query. The user can then navigate through the options and gradually refine her constraints until a suitable option has been found. If multiple options are returned, this refinement process can be very laborious for the user if the system presents options sequentially (an example for sequential option presentation is shown in figure 2.1).

SYS:	*I found 28 round trips from Edinburgh to Brussels on October 31st and returning Edinburgh on November 6th. For option 1, I have an outbound flight with KLM departing at 7 am and arriving at 8:49 am non-stop. A return flight is with KLM departing at 10:30 am and arriving at 11:28 am with 1 stopover. Ticket price is 165 pounds. Please say "next option", or, "flight details", or, "I'll take it".*
USER:	Next option
SYS:	*For option 2, I have an outbound flight with KLM departing at 3:30 pm and arriving at 5:19 pm nonstop. A return flight is with KLM departing at 8:25 pm and arriving at 12:28 pm with 1 stopover. Ticket price is 165 pounds please say "next option", or, "flight details", or, "I'll take it."*
USER:	Next option
SYS:	...

Figure 2.1: Typical Information Presentation Phase of a Communicator Dialogue

The evaluation of nine DARPA communicator systems (Walker *et al.* [2001]) showed that the information presentation phase in dialogue systems accounts for the major part of dialogue duration. Long dialogue duration has been shown to correlate negatively with user satisfaction. Therefore, an objective to several recent projects in the area of spoken dialogue system design has been to shorten dialogue duration and support the user more effectively in finding an optimal option.

2.2 User Tailoring in Generation

2.2.1 Tailored Recommendations

Carenini and Moore [2004] generated user-tailored text recommendations of houses in the context of a multimodal data exploration environment. The focus of their research was the effect of user tailoring in evaluative argument generation. They showed that subjects are more satisfied with the house selections they make if they see tailored recommendations than if they see non-tailored recommendations or just the data without any recommendation, and that they also spend less time making selections if they see tailored recommendations (Carenini and Moore [2001]). Carenini developed two different measures that take into account a user model and are based on argumentation theory and decision theory.
Compellingness is used to identify relevant options that are contained within the option set that matches the user's request. It measures the strength of supporting or opposing evidence for an argument. Options are then selected using the following procedure: the "top" option is determined based on the user model and decision theory. Then the attributes of all other options are tested for compellingness, i.e. whether they are considerably better than the corresponding attribute of the top option. An option which has such a compelling attributes offers a tradeoff to the top options. It is therefore considered to be worth mentioning and selected. *S-compellingness* is a measure for attribute selection. The s-compellingness of an attribute represents the contribution of that attribute to the evaluation of an option (i.e. it mentions why an option is a possibly interesting tradeoff).

An evaluation was conducted, which empirically tested the effectiveness of tailored evaluative arguments on real users. Tailored concise arguments were thereby found to be significantly more effective than non-tailored arguments. The evaluation of the IDEA system (Carenini [2003]) also found a significant impact of argument conciseness on how convincing an automatically generated evaluative argument is.

2.2.2 Tailoring for Spoken Dialogue Systems

Walker *et al.* [2002] applied the decision-theoretic approach introduced by Carenini and Moore (see section 2.2.1) within the multi-modal dialogue generation system for restaurant recommendation MATCH (see also Walker *et al.* [2004]). Here, multi-attribute decision theory was used to enable spoken dialogue systems to focus on the presentation of small sets of options and attributes, that are particularly relevant for the user's decision process.

The effect of user-tailoring and varying verbosity on the quality of automatically generated speech was investigated for three different presentation strategies: summarization, comparison and recommendation. A paired t-test confirmed the hypothesis that people prefer responses generated with their own model over those generated from random models and gave significant evidence for the effectiveness of user tailoring. Evaluation also confirmed that varying conciseness has a strong impact on user satisfaction, indicating that the choice of an appropriate conciseness is crucial.
Evaluation results are also presented in Stent *et al.* [2002], who report a statistically significant positive effect of user tailoring and observe the effectiveness of the comparison and recommendation strategies.

Texts generated by the MATCH system are rather monotone listings of options and their properties (see figure 2.2).

> SYS: *Among the selected restaurants, the following offer exceptional overall value. Babbo's price is 60 dollars. It has superb food quality, excellent service and excellent decor. Il Mulino's price is 65 dollars. It has superb food quality, excellent service and very good decor. Uguale's price is 33 dollars. It has excellent food quality, very good service and good decor.*

Figure 2.2: Utterance in 'comparison'-mode generated by MATCH system

2.2.3 Effective Content Presentation

The FLIGHTS system presented in Moore *et al.* [2004] is one of the systems my work is mainly based on. The FLIGHTS project was an important contribution to the field of spoken dialogue generation because it showed how user preferences could be modeled and used to improve the generation of system responses. The user model affects many aspects of generation, such as content selection and discourse structuring, the choice of referring expressions and the control of prosody. The goal of the project was to generate spoken descriptions that are both memorable and easy to understand. To shorten dialogue duration, the most relevant subset of options was determined and only the most relevant attributes for choosing among them were mentioned. The decision-theory-based relevance measures used in FLIGHTS were the 'compellingness' and 's-compellingness' measures introduced by Carenini and Moore [2004] (see 2.2.1).

To improve structure and understandability of a description, referring expressions that captured the main argument for an option were chosen. Discourse structure was determined by grouping and ordering attributes according to their valuation by the user and discourse connectives were then inserted to contrast positive and negative arguments. This informational structure (i.e. contrast between alternative attributes of an option and their relation to previously presented options) is supported by intonation.
The use of intonation allows one to highlight important points and express contrast intelligibly. In FLIGHTS, theme (the topic of the interaction, on which the attention of the participants is presumably focused) and rheme (new information that is communicated) were assigned to each sentence. The use of theme and rheme annotation and its realization followed former work by Prevost and Steedman (see Prevost [1996] and Prevost and Steedman [1994]). Intonation (i.e. the correct realization of pitch accents and phrase boundaries) conveys the informational structure of the text and thereby supports participants processing of the informational content.

SYS: *You can fly business class on KLM, arriving at four twenty p.m., but you'd need to connect in London. There is a direct flight on BMI, arriving at four ten p.m., but it has no availability in business class.*

Figure 2.3: Tailored description generated by FLIGHTS system

2.3 Stepwise Refinement through Clustering-based Summaries

Polifroni *et al.* [2003] integrated into the restaurant recommendation system MATCH a component that dynamically organizes numeric and symbolic data according to the current dialogue context and available data. The aim was to ease the effort required to adapt a system to new data sets (especially to online content). At each turn, their system calculates a set of options that meet user's constraints and differ along one dimension. Then a summary of the different attribute values of the current option set is generated (i.e. "I have found 19 seafood restaurants. They are predominantly in Back Bay, the North End, South Boston and the South End.", where the user constraint is "seafood" and the dimension along which restaurants differ is their location). The presentation of this summary then suggests to the user a dimension to refine on, but the user may at any time ask about any constraint. So the dialogue flow is predetermined to a much smaller extent than in other systems; it is driven by the user restrictions and the data. This design allows the user to narrow down to the set of relevant options faster and more flexibly than with a traditional option enumerating system.

This approach is implemented using a clustering algorithm that allows to dynamically adjust categories to the data by clustering the attribute values (i.e. calculating clusters of concrete prices which can then be described as "cheap", "low", "medium", etc.). Summaries, that are generated on the basis of these clusters, make use of expressions like "all of them (are cheap)", "none of them (serves seafood)". To achieve maximally effective summarization, the attribute that splits the data into the smallest number of groups is chosen for clustering.

Work on the system has been continued by Chung [2004]. She proposes a constraint relaxation strategy for coping with queries that no option satisfies because they are too restricting. If there is a constraint that can be relaxed (e.g. a certain street could be relaxed to the surrounding neighborhood, or from a Vietnamese restaurant to a Asian restaurant), then an alternative is proposed which includes the next more relaxed constraint that returns a non-empty set of options. When there are several constraints which are not relaxable one of the constraints is deleted. This constraint is chosen such that the remaining constraints produce the smallest set of alternatives.

SIM:	What restaurants do you have?
SYS:	*I have found 983 restaurants. Most of them are located in Boston and Cambridge. There are 32 choices for cuisine. I also have information about price range.*
SIM:	Okay tell me about the ones in Boston.
SYS:	*I have found 401 restaurants in Boston. There are 29 choices for cuisine.*
SIM:	Do you have any that serve seafood?
SYS:	*I have found 19 seafood restaurants. They are predominantly in Back Bay, the North End, South Boston and the South End.*
SIM:	In the South End.
SYS:	*I have found 3 restaurants. Grill Fish on Columbus Avenue, McCormick & Schmick's on Columbus Avenue and Joseph's on High on High Street.*

Figure 2.4: Dialogue between simulator and Polifroni system

2.4 Salience-Driven Text Planning

The work of Chiarcos and Stede [2004] proposes a method to improve local coherence in automatically generated texts. A numerical model for salience is defined, which captures both speaker's intentions and local coherence. The texts generated by the Polibox system are also tailored to users, and the notion of general salience is similar to the notion of compellingness proposed by Carenini and Moore (see section 2.2.1), but in addition makes use of hierarchical discourse structure. The Polibox system generates short description and recommendation texts, where the text structure is determined on the basis of the performance of different options with respect to the speaker's intentions and by contrasting different options against one another (local coherence). An evaluation of the usefulness of the notion of salience is not yet available.

Chapter 3

Conceptual Design

My work combines ideas from two recent approaches (Moore *et al.* [2004], Polifroni *et al.* [2003]). It exploits information from a user model to reduce dialogue duration in spoken dialgoue systems by selecting relevant options and structuring them for stepwise refinement. Thereby, it particularly aims at giving the user a good overview of the available options.

I first discuss the benefits and shortcomings of the FLIGHTS system, namely that it does not propose a strategy of coping with multiple relevant options. Then I address the issue of how relevant options can be identified and analyse the relation between gaining efficiency vs. increasing user confidence.
In section 3.2 I point out the advantages and problems of the Polifroni system and argue how they can be tackled by taking into account a user model. Sections 3.3 concerns the problem of generating coherent texts. The last sections address the issues of increasing user confidence in dialogue systems and improving transferability.

3.1 Content Selection by User Tailoring

The problem with traditional option enumerating systems (see section 2.1) is that it takes a long time for the user to navigate through the option set. The user cannot memorize all the information. Therefore it is difficult for the user to get a good overview of the available options, or understand tradeoffs among options. The FLIGHTS system tackles this issue by selecting the highest valued options with compelling differences

from one another. All options that are determined to be irrelevant using decision theory are not mentioned. The number of options and attributes mentioned can be varied in function of a threshold. But no information presentation strategies have been designed yet for presenting more than three options.
Due to the limited number of option attributes in the flights domain, this method can typically identify a small number of options that are likely to be of interest to the user. This is a size that is easy to manage for the user. In domains where more alternative options need to be considered (for example in the digital camera domain), a strategy for presenting many options has to be developed.
The idea of selecting only the relevant options breaks down the problem into a smaller problem, but does not offer a general solution for presenting a large number of options. An approach that tackles the problem of presenting multiple options more generally was suggested by Polifroni *et al.* [2003], who structure options, summarize them and allow to stepwise narrow down to a small set of options. My approach is to put such a content structuring step on top of a less strict content selection step.

3.1.1 Relevance of Options

The goal of modern spoken dialogue systems is to efficiently support the user in finding an optimal option. Dialogue duration can be shortened, if the options that are irrelevant for the user can be identified and eliminated, so that the user does not have to consider such options. In the FLIGHTS system, as in the work of Carenini and Walker et al. (see chapter 2), relevance of options was determined based on decision theory. Thereby, the relevance of an option depended on the weight of its attributes and their respective values (see compellingness measure Carenini [2003]). All options whose relevance does not exceed a certain threshold are not mentioned.

Originally, my plan had been not to eliminate any options but to structure the options in a tree. The tree would allow for stepwise refinement and keep all options accessible. During the process of constructing target dialogues based on those trees, I discovered that the system would spend a lot of time talking about “bad” options despite the tree structure.
However, I did not want to use the notion of compellingness, because I think that the valuations that are stored about a user in the user model should not make possibly interesting options unreachable. Therefore I defined the notions of “dominant” and

"dominated" options, which is less strict than the compellingness criterion. A *dominated* option is an option that is of no interest for any rational user: it is an option that is equal to or worse in all respects than some other option in the data base. *Dominant* options are all those options for which there is no option in the data set that is better on all attributes. All dominant options represent some tradeoff to one another. Some of them are more interesting tradeoffs than others. Those more interesting tradeoffs (determined based on the user model) are presented first, so that the user first makes the most essential decisions and decides later in the dialogue about the less relevant criteria. The advantage of this ordering is that it minimizes the probability that the user needs to backtrack, i.e. narrow down to option sets in several turns. If an irrelevant criterion had to be decided on first in the decision tree, interesting tradeoffs risk being scattered across the different tree branches.

3.1.2 Efficiency vs. Confidence

One possible solution for not wasting time on dominated options is to not mention them at all. In the extreme case when there is the "perfect" flight which is totally dominant (i.e. better or equal in every single feature than any other flight), the algorithm would only return this one flight. Being informed about just one flight might seem very strange to the user and give rise to the wrong implication that there are no other options available.

Another solution, which I chose for the current version of my system, is to at least briefly account for the bad options. The reasons why an option is dominated are registered during the content selection process and then summarized. Here, a compromise needs to be made between correctness and conciseness. The bad options might have been removed for various reasons, e.g. some because they are indirect, others because they are expensive, or arrive late. In the current version, deleted options are summarized by their most relevant disadvantage (e.g. "all other flights arrive later") and a default recovery "or are undesirable in some other way." instead of "all other flights are either indirect, expensive or late".

3.2 Content Structuring

The solution to the information presentation problem offered in the work of Polifroni et al. breaks the large amount of information down into clusters. The options are structured and sets of options are summarized so that the user can gradually refine her request until she finally narrows down to one option. In Polifronis approach, all options remain accessible to the user and which option she finds in the end depends entirely on her refinement choices. Information is structured according to the user query and the available data, instead of being selected with the help of a user model. In each system turn the set of options which satisfies the current user restriction is presented to the user and additionally, the available options are summarized with respect to one attribute (determined based on optimal summarizability) which may suggest an optional next refinement to the user.
I can see three problems with Polifronis design.

1. **Long paths due to large number of irrelevant options**
   Because the system does not know about the preferences of the user, the option sets contain a lot of irrelevant entities. These need to be filtered out successively with every refinement step. The path to a good option can thus be quite long. Also, the more options there are, the more difficult is it to summarize them. Therefore, such summaries may be uninformative ("I found flights to this destination on 24 different airlines.").

2. **Difficult exploration of tradeoffs when there is no optimal option**
   When there is at least one option that satisfies all requirements, this option can be found efficiently with Polifronis strategy. But if no "optimal" option exists, the user would have to try out different tradeoffs. If there are a large number of equally good tradeoffs, the system does not actively guide the user to these tradeoffs and it might be very laborious to find out these tradeoffs. Consider the following example scenario: the user wants a flight that is cheap and direct, but there are only expensive direct and cheap indirect flights. In the Polifroni system, the user would have to ask for cheap flights and direct flights separately and thus have to explore different refinement paths. For more complex situations, this can become very laborious for the user.

3. **Sub-optimality of procedure which determines the attribute that suggests next user restriction**

> The procedure to compute the most useful attribute which is used to suggest a next restriction to the user, is based on the number of clusters the attribute is split into by the clustering algorithm. If the attribute that splits the data into the smallest number of clusters is not relevant for the user, dialogue duration is increased and the user will be less satisfied with the system, as the my evaluation results suggest (see 8.3.2).

I hypothesized, that the integration of a user model can help to cope with all three problems mentioned above. When a user model is available, it is possible to determine which options and which properties of options are good. The system can then identify those options, that are compelling for the user. The irrelevant options can be deleted from the structure, leading to shorter paths through the tree.
Furthermore, the user model allows to determine, which options represent what tradeoff to one another. These tradeoffs can then be presented explicitly.
Finally, the user model allows to identify at any stage, which attribute is most relevant in the current context. The problem of summarizing a large number of diverse attribute values can be tackled by adapting the cluster criterion to the user's interest.

In my version of the system, I want to allow for an arbitrarily large number of alternative tradeoffs. These would then also have to be structured so that the user can decide for one of the tradeoffs in several steps. For this purpose, a static option tree is built, which contains all dominant options. The structure of the option tree is also determined by the user model in order to ask the user for the most relevant decisions first.
The tradeoffs (e.g. cheap and indirect vs. direct and expensive) are presented to the user explicitly, so that the user won't have to "guess" what tradeoffs might be available in the data base. My hypothesis is that this will lead to a more informed choice and decrease the risk that the user does not find the optimal option for herself. But as the option tree is static, the dialogue will be more predetermined than in the Polifroni system (which can also be seen as an advantage, refer to 3.4).

3.3 Coherence

Automatically generating coherent texts is not a trivial task. For the presentation of several options which represent tradeoffs to one another, it is necessary to mention all attributes that show what the tradeoff consists of. At the same time, the number of attributes mentioned within one turn must not become too large, in order to have a limited number of topics to talk about and to minimize the risk of wildly "jumping" between them.

In the presentation of tradeoffs, ordering the attributes so that an attribute which is mentioned for one option last, gets mentioned for the next option first (e.g. ...but you'd need to connect in Manchester. If you want to fly direct..."') can increase text coherence.

Coherence can also be improved by emphasizing relations between options, for example through contrast (e.g. "X but Y") and expressions relating an option to the other options("the *cheapest* flight", "arrive *earlier*", "not on KLM *either*").

Furthermore, in the refinement process, whenever a set of options is a subset of the option set mentioned before, this relation must be made explicit (for example by formulations like "two of them" "none of them" etc.).

3.4 Gaining User Confidence

There are three types of confidence which can be distinguished in the context of automated information agents.

Firstly, the user should feel confident, that the system correctly returns all options it was asked for. This confidence can be obtained either through experience with the system (the user needs to have used it a lot of times and have experienced, that the system always returns the best options), or the system needs to explain, how and why it came up with the options it presented. The way I chose to do this is by briefly summarizing the options which are not mentioned in detail (e.g. "All other flights arrive later.", "The only direct flight..."). A system that does not at all account for other flights (like the current version of the flight system) has the false implication that there would be no other flights available.

Secondly, the user should feel confident that she found all relevant options. The difference to the first point is that although the user might feel she got all the information she asked for, she might not be certain to have asked the right kind of question. Could the system have offered her other options if only her query had been different? This is a potential problem with systems that let the user guide the dialogue flow. As evaluation suggests (see 8.3.4), this can be a problem within the Polifroni system, where the user

is free to ask for any restriction. In my system this problem does not occur, because at each turn the user is asked to decide to refine on one of the available option sets, of which there is only a small number.
Thirdly, for systems that involve a user model, the user needs to be confident that the information she gave about herself, does not impair the results she gets for her query. The possibility to “overrun” a user model should therefore be given. This has not been implemented into the system yet. For user valuations (relevance ranking of attributes), however, the user model is only used to determine the structure of the option tree and does thus not cause options to be removed from the option tree (contrarily to the FLIGHTS system, where the compellingness measure selects only those options that are particularly compelling on the most relevant attributes).

3.5 Transferability

Clustering attribute values like in Polifroni *et al.* [2003] increases transferability and flexibility of the system with respect to the database and the range of attribute values contained in it. In my implementation, better flexibility is achieved through a mapping algorithm (from attribute values onto numbers) that takes into account the available options, as well as through the clustering algorithm. I did not have time to implement the automated determination of the number of natural clusters, but the final version of this clustering algorithm should allow to cope with any number of natural clusters (for a more detailed discussion see section 5.1.2.3).

Chapter 4

Project Execution

The work aims to present an information presentation strategy that is applicable to many domains. In this project, the strategy was implemented within the FLIGHTS system, a spoken dialogue system presented in Moore *et al.* [2004].

In this chapter, I firstly describe and comment on the flights domain. In the second section, I present the architecture of the generation system. The third section provides an overview of the steps I undertook during the project.

4.1 The Flights Domain

For the implementation of my strategy, the flights domain was chosen, because it was available and required only a modification of an existing system rather than building a system from scratch. Furthermore, using the FLIGHTS implementation as a basis allowed me to reuse the achievements of the FLIGHTS system. These include using referring expressions, scalar terms ("good", "fast", "early") and discourse cues ("but", "though", "if"), as well as a theme/rheme distinction to support the informational structure.

The flights domain is not an optimal domain to show the advantages of my system over the original FLIGHTS system, because it naturally has only few available options (e.g. in comparison to a music cd information system) and few valid tradeoffs; for the flights domain, usually just two or three very interesting options have to be considered seriously.

The options in the flights domain are flight connections (the original FLIGHTS data base included 28 flights from Edinburgh to Brussels on one specific day; for an example data base entry see appendix B). A flight has the following attributes, who contribute to the evaluation of a flight: “arrival-time”, “departure-time”, “number-of-legs”, “travel-time”, “price”, “airline”, “fare-class” and “layover-airport”. A user model contains a partial ordering of these attributes that correspond to the user’s valuation. Furthermore, preferences and dispreferences (e.g. preference of a special airline or for flying on business class etc.) are stored in the user model. In a real-world application scenario, the user should register with the system once. When she uses the system, she only needs to specify actual situational information such as the destination and desired arrival time and date. There are three exemplary user models. They include

- a student who cares most about price, all else being equal
- a frequent flier who collects business miles on KLM and therefore care most about airline
- a business traveler who wants, above all, to travel in business class and prefers also KLM

The business traveler user model is given in appendix A.

4.2 Architecture of Text Generator

Text generation in the system makes use of the FLIGHTS system pipeline architecture. The first step in generation is the *content selection* step, (also called content determination in Reiter and Dale [1997]). Its function is to decide which information should be communicated in the text. This generally includes the filtering and summarization of input data. In my system, the input to the content selection step consists of the data base, the user model and the current user query. The content selection component is written in Java.

Content planning is the next step. The task of this step is to impose an ordering and a structure on the content that is to be conveyed. Ordering messages can enhance understandability and coherence of a text (for example a special structure for identifying

an option as opposed to elaborating an already introduced option). Positive and negative aspects of an option can for example be contrasted against one another. Strategies that aim to manipulate user choice (e.g. by ordering bits of information to increase or decrease their effectiveness) could also be implemented in this part of the generation pipeline.
The content planner used in this system is implemented in an Lisp-based AI planning language (O-PLAN), that uses schemas.

As the output from the content selection step is not in a form that can be directly processed by the content planning module, an interface component translates the xml-tree produced by the content selector into a LISP structure, using XSLT.
The resulting content plan is translated and can then be read in by the sentence planning component. The sentence planning component in this system uses templates to perform lexical choice, aggregation, and to build up alternative logical forms. These logical forms are combined into a single packed representation, which contains alternative formulations. The packed representations are sent to the CCG-based realizer, which transforms the logical forms to natural language sentences and makes the final choice on structure using a statistical n-gram model.

In the system presented here (as well as in the FLIGHTS system, which my system is based on), the content planner enriches the content plans with information about theme and rheme. The sentence planner selects sentence structures depending on this theme/rheme distinction and the realizer produces the sentences with a suitable APML[1]-annotation. Speech with correct pitch accents and boundary tones can then be synthesized using Festival[2].

The focus of this project is on the content selection and content planning steps. Therefore, the speech recognition and speech synthesis components of the FLIGHTS system were not adapted. Communication with the system currently works via a text input/output agent. Also, I did not have time to integrate a parser into the system. My system uses the FLIGHTS keywords spotting component and a very simple interaction for indicating user choices.
The different components are implemented as agents in the OAAAgent framework

[1] Affective Presentation Markup Language
[2] http://www.cstr.ed.ac.uk/projects_festival/apml.html

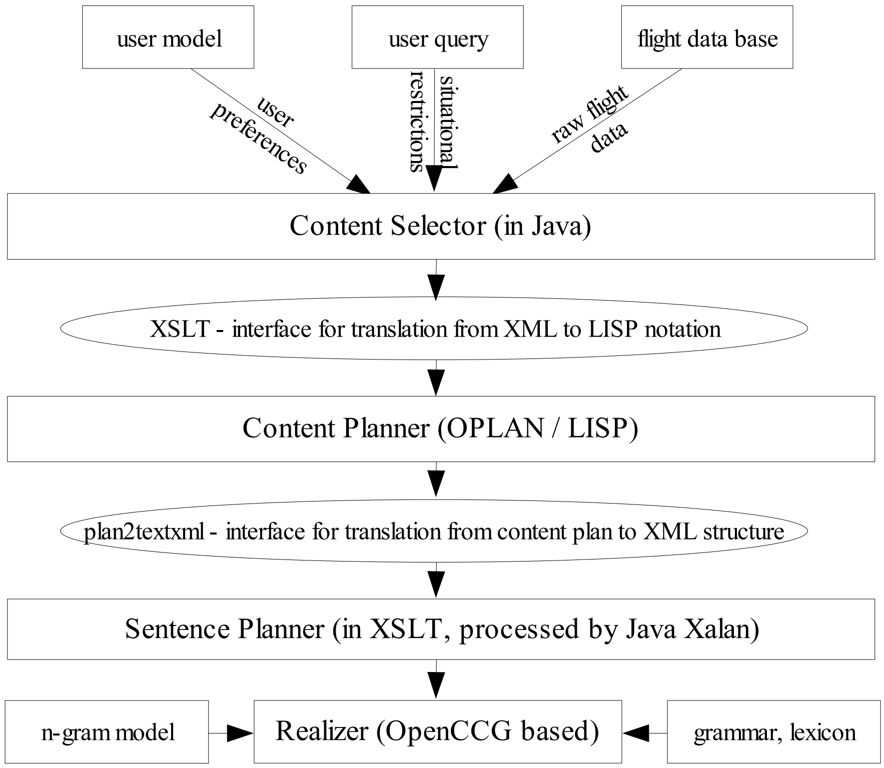

Figure 4.1: System Architecture (Text Generation Part only)

Martin *et al.* [1998] as a communication hub. All modules are implemented as agents, whose communication is managed by the DIPPER dialogue manager agent, that calls the different agents and stores the intermediate results from each component.

4.3 Main Steps in Project Realization

A research review of the field and a project proposal were completed during the taught semesters. The practical work on the project and dissertation write-up were done within a 13 week time frame.

In the first phase of the project, I made myself familiar with the implementation of the FLIGHTS system in order to find out how to use the available structures effectively and how to integrate my new algorithms. During this phase I conducted a feasibility study for the adaptation of each of the components. The biggest changes had to be done in the content selection component, within which the core clustering algorithm and option tree construction would be carried out. Extensive modification of the content planner would also be necessary in order to allow to talk about sets of flights instead of single flights. All subsequent components (the text2planxml interface, the sentence planner and the realizer) would need to be adapted accordingly, but would not require creative work.

Altering the FLIGHTS' user models was not necessary, but in order to test the new clustering algorithm and to show transferability to new data entries, the data base was extended and includes now a wider variety of flights and attribute values.
Next, the evaluation algorithm that maps attributes of different types (such as prices, times or categorical values like airlines) onto numbers was reimplemented in order to allow for transferability to new data entries. In the FLIGHTS implementation, the data base entries were not taken into account for the mapping, e.g. all prices above £1000 were mapped to 0.
A clustering strategy (agglomerative clustering) was chosen and implemented with a static number of clusters, the automatic determination of the appropriate number of clusters was left for later optimization. Unfortunately, I did not have time to come back to this. An option tree was constructed based on the clusters and on the attribute valuations defined in the user model. While constructing a first set of concrete target dialogues based on the data, I discovered the difficulty of creating summaries that include options which are of no possible interest for the user. These first dialogues sounded very unnatural and consumed too much time for talking about irrelevant options. From this observation, the concepts of "dominance" (an option is dominant if it is equal or better in all respects than another option) and "justification" (each option that ends up in the pruned cluster tree must be justified by a property that makes it particularly compelling with respect to the alternative options) were developed and the additional pruning step was implemented.
A very simple strategy for determining turn length was then implemented: the pruned option tree was cut into chunks that comprised two option tree levels and all attributes that were homogeneous for some node in a chunk. The content selection agent was extended to access and return the different tree chunks, keyed on the path toward them in the tree. I came back to the turn length issue later on in the project and implemented a more sophisticated method, which matched the target sentences better than the first simple implementation.

These steps had put the core algorithm into place. Now, the xml output of the content selection step had to be translated into LISP structures, so that the content planner could read it in. The content planner uses the AI Plan language OPLAN and performs the content planning based on templates. The satisfaction of conditions for the application of a template and its contents are processed by LISP functions. Within this module, the data structure had to be adapted in order to process the recursive option tree. New functions were provided to retrieve option justifications which are used for the identification of an option. Due to the fact that an option tree node might either contain sets of options or a single option, a summarization strategy had to be implemented. This summarization strategy allows for user-tailored summarization of flights (e.g. "not on KLM", "arrive earlier", "connect in Amsterdam or Frankfurt"). Furthermore, keywords that evaluate an option (such as "cheap", "fast" , "early" etc.) are decided on in the content planner, and functions for determining these evaluations and making them accessible for the templates had to be written. Finally, a new function for user query refinement, i.e. navigating in the option tree, had to be added to the agent functionality. This function can be called by the dialogue manager and has the task of processing the user choice and sending the corresponding query to the content selector.

The content plans produced by the content planning module need to be translated into xml-trees. This is the task of another Java-based agent called "Plan2TextXml". This agent had to be adapted in order to be able to transform the summary structures "between", "and", "or" and "not". Once the exact formulations for the target dialogues had been finalized, a set of target sentences was sent to Michael White who extended the realizer accordingly. His work provided me with the output specification for the sentence planner.

Then the sentence planner was adapted so that it could process the input from the Plan2TextXml agent and meet the output specification for the target sentences.

Finally the dialogue manager, that coordinates the tasks of the agents, was modified, so that it returns a refinement choice to the user and allows for several user-system turns.

To assess the merit or failure of my approach, a within-subject experiment was designed with E-Prime to evaluate the system against the system presented in Polifroni *et al.* [2003] and Chung [2004]. The experiment involved 38 participants who answered questions to 10 dialogues, of which one half was produced by my system and the other half were designed according to the Polifroni system. The results were analysed, tested for statistical significance and interpreted.

Chapter 5

Content Selection and Structuring

This chapter first presents the attribute clustering step. I describe the necessary pre-processing step (which consists of mapping attribute values onto numbers), summarize different possible clustering strategies, justify my choice for a clustering algorithm, discuss the problem of determining the number of clusters and finally comment on the general of clustering attribute values for summarization.

The second section of this chapter concerns the process of building up a tree structure based on the clusters and explains how the user's valuation determines the structure.

The tree pruning algorithm, which removes all options that are irrelevant to the user from the tree structure, is described in detail in the third section of this chapter. I also present the important by-product of the pruning algorithm, which consists of the identification of what makes each remaining option an interesting tradeoff.

Finally, I discuss turn length determination and explain my method of attribute selection.

5.1 Clustering the Objectives

In order to build up the clusters for each attribute, the attribute values (having different types such as prices, times, airline names etc.) must first be mapped onto numbers.

5.1.1 Mapping of Attribute Values onto Numbers

5.1.1.1 The Original Version

For the original FLIGHTS system, a function for mapping attribute values to values between 0 and 1 is stored in the user model. Discrete values (like airline: KLM) are mapped onto evaluations based on user preferences. For example the value KLM would be mapped to 0.8 if the user had marked it as her preferred airline, to 0.5 if she does not care and to 0.2 if she dislikes it. Storing this mapping in the user model has the advantage of high modularity and easy modifiability (for example to readjust the scale to specific users or to introduce a finer scale adapted for a certain user).
A problem is that these evaluation functions only take into account the user model but not the content of the data base. For the continuous values of attributes like arrival-time, travel-time or price, it is important to scale the value in dependence of the other flights available. (Consider for example a data base, where prices for a certain query range between £50 and £150 in comparison to one where prices range from £200 to £2000.) The current implementation uses fixed functions like `Math.max( 0.0, 1.0 - ( price / 1000.0 ))` for the price attribute, which causes all mappings for flights that are more expensive than £1000 to be meaningless.

5.1.1.2 The Issue of Transferability

One goal of Polifroni *et al.* [2003] was to create automatic methods that are more robust against the frequent changes to on-line content and make the adaptation to new databased less laborious. They proposed the use of data-driven clusters to organize numeric and symoblic data and implemented this approach to the restaurant recommendation system MATCH. Each cluster is assigned a label such as “cheap”, “expensive”, “near” etc. dependent on the data base entries. A restaurant that was 3 km away could then be described as “near” for a rural area with widely-spread few restaurants or “far” in a city center where a many restaurants is much closer.

To support transferability to new data sets, I decided to move the mapping from the user-model data structure to the content selector module, where it can be driven both by the available options and by the information from the user model. The effect of this modification (described in 5.1.1.3) was tested by generating flight information for flights with properties which are different from the original data base entries (for example having a different price range and offering only indirect flights).

5.1.1.3 Modification

When objectives have categorical values (like airline: like vs. dislike vs. don't care), the mapping from the user model (0.8, 0.2, 0.5 respectively) is maintained, so that easy modifiability is preserved.
The continuous values (times and prices) need to be type-converted to numbers between 0 and 1 which represent the goodness of the attribute value with respect to the user model. For prices, a value independent from the currency is determined and rescaled dependent on the price range of options that satisfy the current query. Times are converted to a minute notation that reflects the difference between actual and desired arrival / departure times. Travel times are calculated from the difference between the departure and the arrival times, a problem being the time lags. A time zone database would have to be included into the system, which has not been done yet. In the current implementation, time lags can only be calculated correctly for those destinations, to which there are entries in the data base.

5.1.2 The Clustering

In the FLIGHTS system, a value is labeled "positive" or "negative", if it deviates more than one standard deviation from the average value for that attribute. In my implementation, following the example of Polifroni Polifroni *et al.* [2003], the labels are assigned to the clusters determined during the clustering step.

5.1.2.1 Clustering Algorithms

"Clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure."
(wikipedia's definition of clustering, http://en.wikipedia.org/wiki/Data_clustering)

There are two main techniques for clustering: partitional clustering and hierarchical clustering. *Partitional clustering* has the advantage of typically having a lower space and time complexity. For partitional clustering, the number of clusters has to be specified beforehand, all clusters are then determined simultaneously. The best known partitional clustering methods are k-means clustering and the EM algorithm, which is well suited to determine clusters in multidimensional data.

Hierarchical clustering successively determines clusters on the bases of established clusters. That's why a decision once made cannot be undone in subsequent clustering steps. There are two ways of doing hierarchical clustering: either bottom-up (agglomerative clustering), or top-down (divisive clustering). In *agglomerative clustering*, which is the by far more common method, the algorithm is initialized with each element as a separate cluster and merges them until some stopping criterion is met. In *divisive clustering*, a set that initially contains the whole data set is successively split into smaller clusters. A precondition for hierarchical clustering is that the similarity function which is used to determine whether clusters should be merged (in agglomerative clustering) or separated (in divisive clustering), is monotonic, such that the merging operation is guaranteed not to increase within-cluster similarity.

In hierarchical clustering, there are different methods for determining the similarity of two clusters. *Single-link clustering* merges those clusters, that have the highest similarity between their most similar members. The criterion for merging in *complete-link clustering* is the similarity of the two most dissimilar cluster members, and *group-average clustering* chooses the two clusters with the biggest average similarity between their members. Among these methods, single-link clustering most easily leads to very elongated clusters or "chains", which is an undesirable effect in this application, where the goal is to obtain compact clusters. For a more detailed discussion on clustering see Jain *et al.* [1999] or Manning and Schütze [1999].

5.1.2.2 Chosen Clustering Algorithm and its Implementation

For my data, space and time complexity is not a relevant issue, because for the flights domain, there is only a very limited number of data points. Also, I don't need a very fancy method that is optimal for multidimensional problems, because the data points are only one-dimensional, as clustering is done for each attribute separately.

I decided to use agglomerative clustering for my problem, because it is easy to imple-

ment and can be extended to determine the optimal number of clusters (this problem will be discussed in section 5.1.2.3). As an initialization, all flights with the same value for an attribute are collected in the same set. For the continuous valued attributes (e.g. arrival time), there may be as many sets as flights, whereas the discrete valued attributes (e.g. fare-class) partition the flights into natural clusters (e.g. flights with availability in business class and flights without availability in business class).
Similarity between clusters is determined using the group-average clustering criterion, which merges the two clusters with the most similar averages.

5.1.2.3 Determining the number of clusters

In the current implementation of the system, the default number of clusters per attribute is set to three. For the attributes with categorical values, there may only be two clusters (in the data for flights to Brussels for example for number-of-legs only "1" or "2" and for fare-class only "business" or "economy"). The reason for setting the stopping criterion for the clustering algorithm statically to three clusters is that it becomes increasingly difficult to summarize the data and to talk about it, if there is a larger number of clusters. Three categories are assigned to the clusters: "good", "bad" and "other". More fine-grained distinctions are currently not supported, even though this restriction can be very problematic, as I will explain with the following example: Within the set of flights that go to Brussels, prices fall into four natural clusters: The first contains flights that cost around £50, the second contains flights for around £100, the third for £200 and the last one flights for £400. When these prices are mapped linearly onto numbers between 0 and 1, the two cheapest categories are clustered together. This leads to the problem that the difference in price (between £50 and £100) will not be regarded relevant any more. In the pruning step (refer to 5.3), all £50 flights might then be deleted and thus not be mentioned to the user. This is totally unacceptable for a user who cares for price. Even if the mapping from prices to values between 0 and 1 was not linear but based on the relative differences between prices, the problem would not be solved for the above example.

There are two possible solutions to this problem: One can introduce a more flexible number of clusters (such as "flights with a very good price", "flights with a price better than average", "flights with a moderate price", etc.) despite possibly lengthy summarizations, or clusters can be combined based on data from the user-model. Then, it could be argued that for a user who cares for price, both the £200 and the £400 flights would be unacceptable and could thus be clustered together.

Now that I have motivated why it is important to be aware of the number of clusters the data falls into naturally, I want to explain how the number of natural clusters can be detected. If the data falls into a certain number of k clusters naturally, then it is generally possible to observe a substantial increase in goodness in the transition from k-1 to k clusters and a small increase in the transition from k to k+1 clusters. In order to automatically determine the number of clusters, we can look for a k with this property and then settle on the resulting k clusters (from Manning and Schütze [1999]). Here hierarchical clustering proves useful: the values for different numbers of clusters k can be determined while running the algorithm. In the case of partitional clustering on the other hand, it would be necessary to rerun the whole algorithm for all numbers of clusters that should be tested. A more mathematically founded approach to the problem would be to use minimum description length to determine the optimal number of clusters.
For any cluster determination method, it is necessary to define a quality measure for the clusters. In this application, such a quality measure could consist of the variance within the cluster and the difference in similarity between different clusters, as well as an integration of what the user cares about as known from the user model.

5.1.2.4 Discussion

The current implementation uses the group-average clustering criterion. Unfortunately, this criterion does not optimize within-cluster similarity and inter-cluster dissimilarity actively, because variance within the clusters is not considered. I conclude that there would be much room for optimization on the clustering algorithm, which I did not have time to implement, as the project's focus was not on optimizing clustering algorithms but on the investigation of the benefit of taking into account a user model for summarizing options based on clusters.

Another issue is the appropriateness of clustering attribute values in general. For some attributes in the flights domain, like price, values fall into clusters naturally, because price is mainly dependent on the airlines operating the flights and how well they cover the itinerary in question. In other cases, values might just not naturally fall into clusters, in the flights data this is the case for the arrival time. Arrival times are quite evenly spread across the whole value range. There is no natural reason to put them into clusters. Therefore, assignments to clusters and inter-cluster borders risk to seem arbitrary.

5.2 Building the Tree Structure

The nodes of an option tree correspond to sets of options; the leaf-nodes correspond either to a single option or to a set of options with the same properties. The children of a node are subsets of the parent option set and complementary with respect to one another. Each child is homogeneous for the same attribute (for example if the parent set includes all direct flights, its first child might include all direct cheap flights and its second child all direct expensive flights). At each level, the attribute, which is most important for the user in order to present the user with a relevant choice is chosen. So the branching criterion for the first level of the tree is the attribute that has the highest weight according to the user model.
A node stores all attributes that are homogeneous for its set of options. There is always at least one homogeneous attribute per node (the one that was the criterion for creating the node) but there might be several if the options in a set in a node are homogeneous with respect to several attributes.

To achieve a small tree (because a small tree is heuristically easiest to summarize and requires the minimum number of decisions and reduces dialogue length), at each such step, a search over all attributes with rank one lower than the actually considered rank is done to see if the current subset of flights is homogeneous for an attribute. In that case, the homogeneous attribute becomes the next (unary) node regardless of its rank. This has happened in the example tree in figure 5.1 in the right branch, when the unary node with attribute “airline” is inserted.
As the user is allowed to say that two attributes, e.g. arrival time and price, are equally important to her, attributes will only be partially ordered in most user models, which leads to several attributes having the same ranking. The algorithm determines into how many clusters each of the equally ranked attributes split the current set of flights and chooses the attribute that splits the data into the smallest number of sub-clusters. This

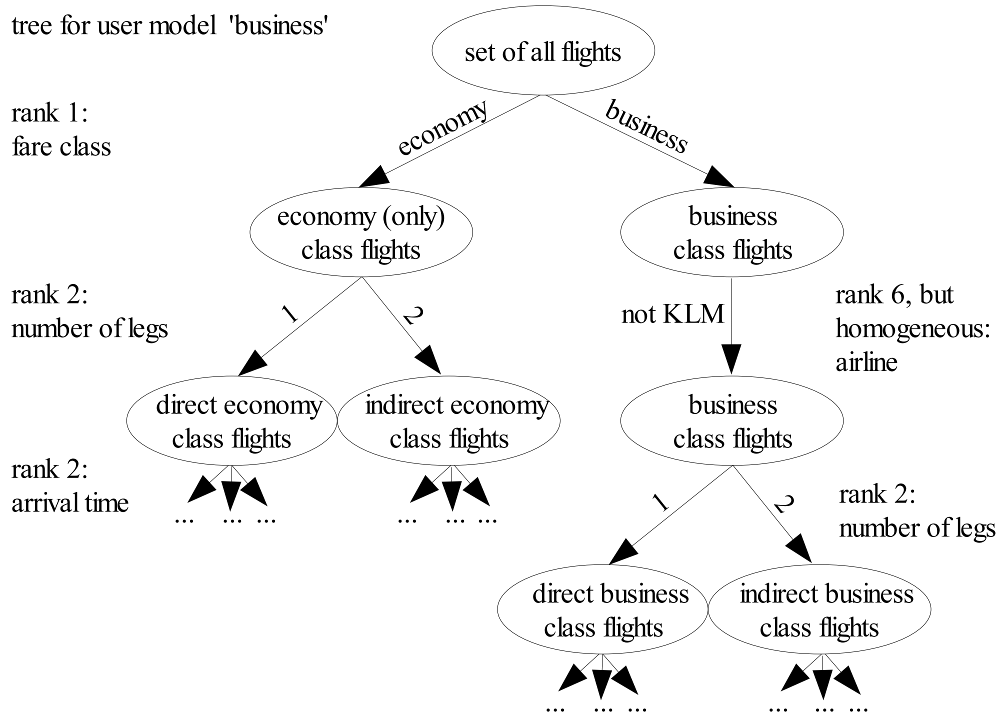

Figure 5.1: Building up the Option Tree

strategy was also used in Polifroni *et al.* [2003] to determine the attribute that keeps summaries optimally short and efficient. For illustration see again the tree in figure 5.1 where “number-of-legs”, having two sub-clusters for the actual set, comes before “arrival time” which would split the current set of flights into three sub-clusters.

5.3 Pruning the Tree

Pruning the tree is the first content selection step, in which the choice of flights to be described in detail is made. The second content selection step (described in section 5.5) concerns the choice of attributes that are mentioned to describe each flight. This step is parallel to the determination of compelling entities in the FLIGHTS system.

A *dominant* entity is an entity such that it is not equal or worse in every respect than some other entity. All dominant entities represent some sort of tradeoff to one-another. Finding the dominant entities can be very inefficient if done naively, because it requires to compare every aspect of every flight to the corresponding aspect of every other flight and keeping track of these comparisons in order to find pairs of dominant-dominated

entities. In this work the tree structure is exploited to increase efficiency (see algorithm in 5.3.1.6).
Leaves in the pruned tree may contain more than one option, because the construction and pruning of the tree is based on clusters (rather than on the original values). Options that happen to be in the same leaf are very similar, as they are in the same clusters for every attribute. For the same reason why bad options are pruned, it is not interesting to present all of these similar options to the user. Therefore, the best out of these similar options should be determined on the basis of the original values and only the best option should be mentioned. This further leaf-pruning step has not been implemented yet, due to time limitations. For the flights domain, this case of many similar options is very rare.

In the following sections, I explain the pruning algorithm with the help of a concrete example, illustrated in figures 5.2 to 5.7.

5.3.1 Overview of Pruning Algorithm

The input of the algorithm is the ordered cluster tree. This means that for each node, the order of its children is rearranged such that the "best" child always comes first, followed by the next best child and so on (see illustration in figure 5.2).
This horizontal ordering guaranties, together with the vertical ordering which reflects

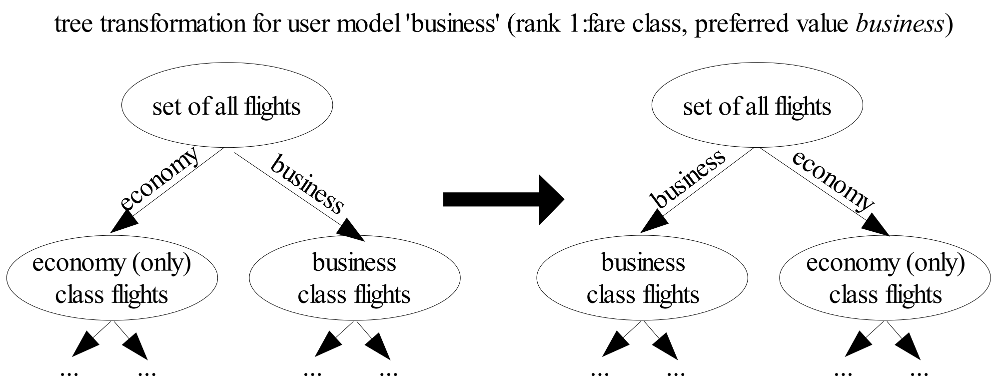

Figure 5.2: Ordering of Nodes in Option Tree

user's valuations, that the "top" flight is in the leftmost branch of the tree. The main idea in the pruning algorithm is that all those entities that are in branches to the right of the currently considered node are dominated by it, unless they have a good value for at least one of those attributes, which the current node has a bad value for. A node that is optimal (which has the highest possible value with respect to all objectives for which its flights are homogeneous) does not generate any constraints. The right siblings of an optimal node and their children don't need to be considered, because they are known to be dominated by the entities in the current node. Why is this? Two sibling nodes share all attribute values that are described in the upper layers of the tree. Because of the horizontal ordering, they must be worse in all respects than the left node. The argumentation is the same when the left node generates constraints which right siblings cannot satisfy.

5.3.1.1 Generation of Constraints

The algorithm transverses the tree in in-order. It goes down to the leftmost leaf and generates *constraints* in case the flights of this node have some negative attribute. A constraint is a condition that must be satisfied by all right siblings of a node. For example, if all flights in a node have a price that is in the cluster of expensive prices, then the constraint generated from this would be that the value for the "price" attribute must be at least average (or whatever the next better category is). All direct siblings must satisfy the constraint, otherwise they are pruned. For illustration see figure 5.3, which does not show the flight entities to save space.

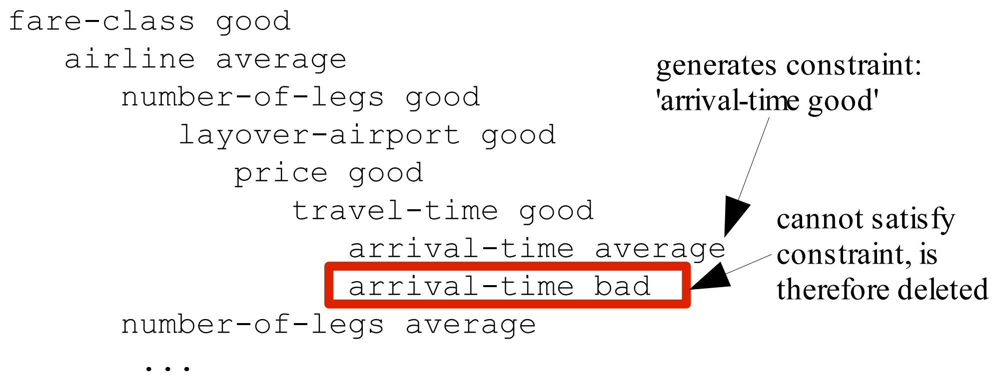

Figure 5.3: Pruning a Sibling

5.3.1.2 Constraint Propagation

If a node generates a constraint, this constraint is *propagated up* in the tree, in order to find an entity that satisfies it. In figure 5.4, the node "number-of-legs good" imposes the constraint "arrival-time good", which was propagated up from its child, on its sibling "number-of-legs average". (The other child was deleted and can thus not generate constraints.) For the sibling node, it is currently undecidable, whether it can satisfy the constraint. Therefore, it *inherits* the constraint to each of its children, and they inherit it further onto their children, until a node is reached that has the attribute "arrival-time" within its set of homogeneous attributes. In the example, just one child can satisfy the constraint, the other children "arrival-time average" and "arrival-time bad" are removed without exploring their child nodes.
When all the children of a node are deleted because none of them satisfied the constraints, the node itself is also deleted (it represents no more entities).

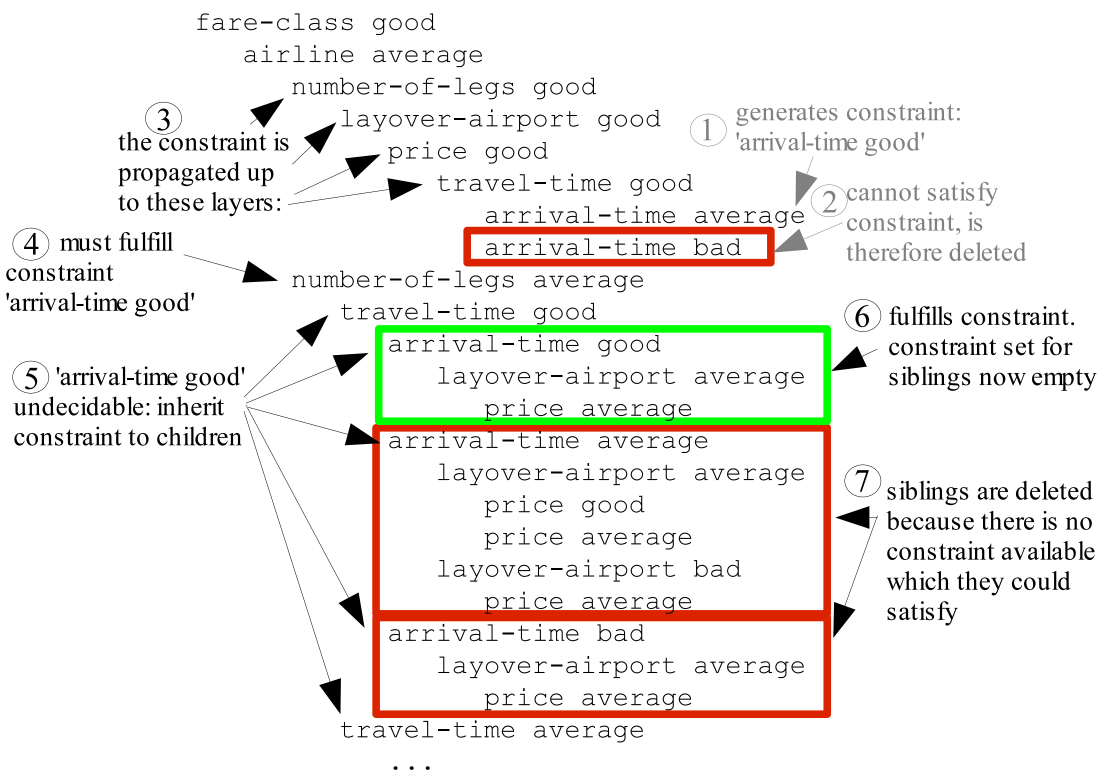

Figure 5.4: Steps in Pruning Algorithm I

5.3.1.3 Constraint combination

The cluster that survives the pruning removes the old constraint from its constraint list (because it has satisfied the constraint) and adds new constraints which it generates on the basis of its homogeneous values, as explained above. In case a node has still at least one child after the pruning procedure, the constraints generated by its children are combined with the constraint generated by the node itself.

In the example, two constraints are generated ("travel-time good" and "layover-airport good"). Any sibling has to satisfy either of these. The new constraints are then propagated up to the parent node again and applied to the node's siblings ("travel-time average" and "travel-time bad").

There are now two sets of constraints applying to these nodes: the ones that were inherited from the parent node and the ones that were generated by the left sibling. These are combined as follows: Each node now has to satisfy the inherited constraints and the constraints from the left sibling, leading to the constraint: ("price good" or "layover-airport good") and "arrival-time good". Nodes are deleted accordingly. For illustration see figure 5.5 and 5.6.

Constraints may get very complex when constraints from left siblings are combined with constraints inherited from parents: they are combined by generating all permutations of inherited constraints with constraints from siblings.

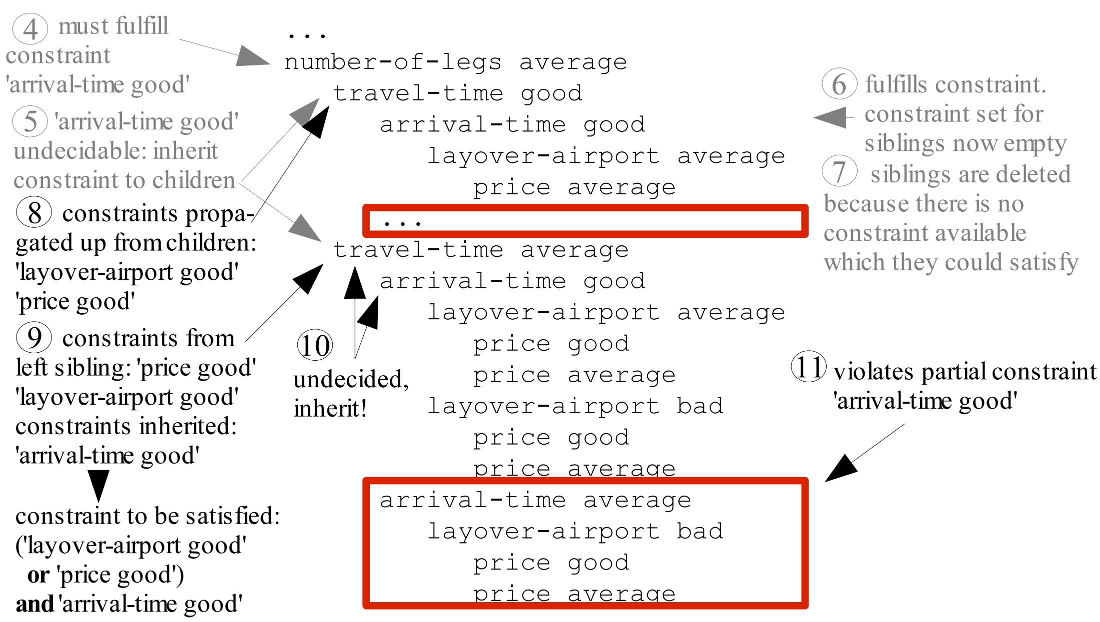

Figure 5.5: Steps in Pruning Algorithm II

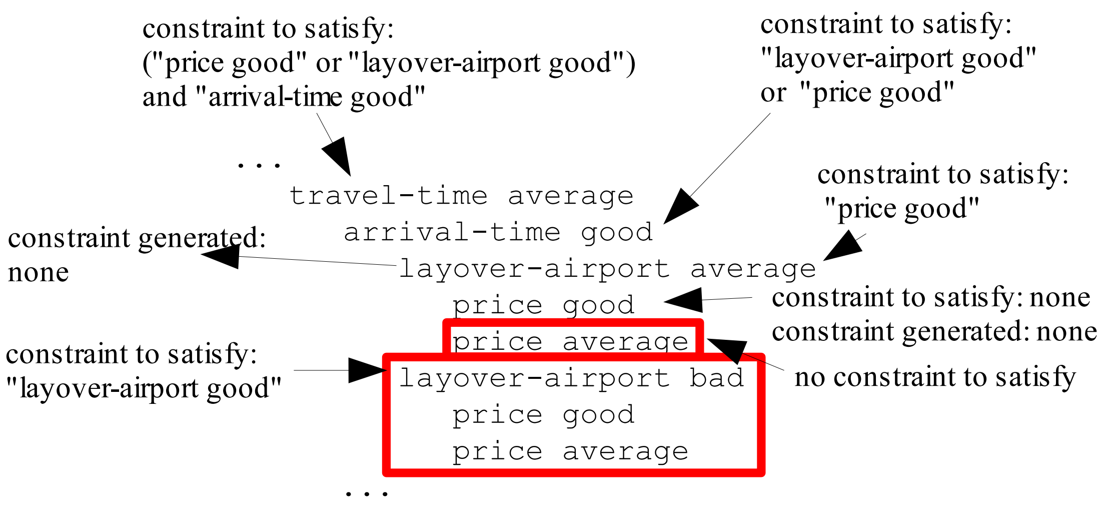

Figure 5.6: Steps in Pruning Algorithm III

5.3.1.4 Justifications

The pruning result (see the full length unpruned tree and its pruned version in appendix C) is a much slimmer structure that contains only valid tradeoffs. (In the extreme case this might just be a single option.) The algorithm stores the *justifications* for cluster that has been preserved. These justifications can be actively used for referring to the clusters in order to point out what makes them compelling. For a more detailed discussion of justifications see section 5.3.3.

5.3.1.5 Tree Readjustment

A special case is when two or more children survive the pruning step. If they both have the same value for one attribute, this attribute is homogeneous for the parent node and the tree thus has to be readapted to reflect this fact. Therefore, the tree is modified, such that the respective unary node that represents the attribute attribute is deleted from each of the children and instead inserted as a unary node under the parent node. Special care has to be taken, to correctly assign justifications to these nodes. The attributes that are move up in the tree, are removed from the justification of the nodes and instead added to the justification of their parent node.

```
number-of-legs average
    travel-time good
       arrival-time good
          layover-airport average
             price average
    travel-time average
       arrival-time good
          layover-airport average
             price good
```

```
number-of-legs average
  arrival-time good
     layover-airport average
        travel-time good
           price average
        travel-time average
           price good
```

Figure 5.7: Tree Readjustment after Pruning

5.3.1.6 Notes on Implementation

For efficiency, the algorithm uses a compact representation of the cluster tree in which all unary nodes are combined to a single node and their homogeneous attributes are stored for that node. So the algorithm always steps to the next branching layer and controls satisfaction of constraints for all homogeneous attributes at the same time. Here is the main algorithm in pseudo-code:

```
for i = 0 to i <children.size do
  child ⇐ children(i)
  result ⇐ satisfiesConstraint(child, constraints)
  if result ⇐ constraint satisfied then
    recursively apply algorithm to child's children, inherit no constraints
    child_constraints ⇐ constraints generated by child's children
  else if result = constraint violated then
    delete and continue
  else if result = currently undecidable then
    recursively apply algorithm to child's children, inherit undecidable constraints
    child_constraints ⇐ constraints generated by child's children
  end if
  if all children were deleted then
    delete self and continue
  end if
  if attribute has become homogeneous through the effects of pruning then
    fix tree
  end if
  combine child_constraints with inherited constraints
end for
```

In the following sections, I will give a more detailed description of the pre- and post-conditions of the constraint satisfaction checking and explain the registration of cluster justifications.

5.3.2 Satisfaction of Constraints

One can distinguish between three kinds of constraints:

- constraints inherited from parents
- constraints from left siblings
- constraints generated by self or propagated up from children.

For the first two points, a distinction between leftmost and non-leftmost child is necessary, because the constraint set being empty means for the leftmost child, that it will be accepted no matter what but for the other nodes that there is no constraint which they can satisfy. An alternative design to avoid this case distinction, would be to first generate a set of all possible reasons (permutations of all attributes and all their possible values) and eliminate from this set all those which are satisfied. This would be very computationally and an memory expensive.
Algorithm for checking the satisfaction of constraints:

if constraint set is empty **then**
  if node is leftmost child **then**
    accept node and generate constraints
  else
    remove because no constraints available
  end if
else
  for $i = 0$ to $i <$size(constraintConjuncts)[a] **do**
    actualConstraint $\Leftarrow$ constraintConjuncts(i)
    check satisfaction of actualConjunct[b]
  end for
  if each constraint conjunct is violated **then**
    return 'violation'
  else if all constraints from at least one constraint conjunct are satisfied **then**
    return 'satisfaction'[c]
  else if the constraints of at least one constraint conjunct are not all violated **then**
    return 'undecidable'
  end if
end if

[a] A condition has the form:
$((a_1 \vee a_2 \vee ... \vee a_n) \wedge (b_1 \vee b_2 \vee ... \vee b_n) \wedge ... \wedge (m_1 \vee m_2 \vee ... \vee m_n))$

[b] For efficiency, the constraint conjunctions are ordered with respect to their lengths. This also has the advantage of generating the shortest possible reason, i.e. the justification for the cluster that involves the smallest number of disjunctions and conjunctions.

[c] The constraint set is updated such that the next sibling cannot be accepted for satisfying the same constraint but only those which the actual cluster does not fulfill. Any constraint that is fulfilled by the cluster is added to the set of reasons for justification of the cluster.

5.3.3 Registration of Justifications

Each cluster needs to be identified by its attribute that is most relevant for the user's decision process. With this perspective, the attributes that justify the existence of a node in the pruned tree are collected and held available, because these attributes represent the tradeoffs between alternative options. Again, there are three different cases:
If a node survives the pruning step because

1. the constraint set was empty and it was the leftmost sibling
   Then its justification is its position as the leftmost from the ordered siblings. The cluster criterion (the homogeneous attribute from the uppermost node, if there is a chain of unary nodes) is the justification for this cluster.

2. the cluster satisfies all constraints
   If a cluster satisfies a constraint, all attributes that satisfied the constraint are registered and added to the set of reasons. To allow for effective summarization and reference to clusters, the constraints are sorted from shortest constraint to longest (this increases also efficiency of the algorithms). By simply changing the sort algorithm to descending length (which would make the test for constraint satisfaction less efficient), exhaustive reasons can be generated.

3. its children satisfied the constraint(s)
   A cluster may not have a reason of its own (for example because it has just homogeneous attributes that are the same or worse than the ones of the option it is contrasting with). I then gets assigned a disjunctions of the justifications of its child nodes. A cluster that inherited all reasons from its children has thus a disjunctive reason.

In order to account for the whole dataset (to tell the user that the whole option space was explored) short summarizations about flights that were pruned off should be given. These flights which were sorted out should be summarized by their reason for being deleted. Similarly to the registration of justification for the entities that persist in the structure, the reasons for clusters to be deleted are also collected and stored with the surviving sibling.

5.4 Turn length

In a spoken dialogue system, it is important, not to mention too many things in one turn in order to keep the memory load on the user manageable. Even the pruned tree has, dependent on the user model and query, still three[1] levels of depth for many queries in the flights domain. That means that three decisions are required, before a leaf node is reached. If too many depth levels are presented in on turn, the turn gets very long and potentially very complicated, because too many different tradeoffs are presented. It will then be difficult for the user to keep track of the structure and the different options available. For example with a branching rate of just 2 and a depth of 3, the user would have to remember $2^3 = 8$ options and their attributes. If applied to other domains, the number of levels in the pruned tree might still be higher.

Obviously, not all of these choices and options can be presented on one turn. It is necessary to cut the tree into several smaller trees that can then be presented over several turns. One possibility is to cut the tree level-wise. But then the information would be too brief and the choice trivial if there is just one homogeneous attribute: "Do you want to fly direct or make a connection?" "Do you want the cheap flight or the expensive one?" These questions are ridiculous, because the answer is already known to the system through the user model. They are not capable of representing the what constitutes the tradeoff between alternative options. Furthermore, too many turns increase the dialogue duration, which is known to have a negative effect on user satisfaction (see section 2.1).

All attributes that are relevant for what makes the option set a tradeoff with respect to an alternative set of options need to get mentioned in one system turn. Otherwise the user cannot make an informed decision. If the flights in a node represent e.g. the following tradeoff: "direct but either expensive or arrives too late", it is not only necessary to talk about one layer in the tree ("direct") but also about its children (which contribute the one "expensive" and the other "too late").

To give the user a good basis for decision, the disadvantage of an option and in how far an option is compelling must be presented. In general, the explanation depth should go down to a level where the attributes that justify the option are homogeneous (and thus easy to summarize).

In the current implementation, a heuristic cut-off point is used. Two levels are presented in one turn. This guaranties to produce a small set of options to talk about

[1]not counting unary nodes, but only decision levels

in a turn (for actual data in the flights domain usually three to four) and includes the main relevant advantages and disadvantages of an option. For an information sequence where the one-level turn length would have yielded[2] *"There are four flights with availability in business class, but none of them are on KLM. I also found some flights that only have availability in economy class."*, a cut-off after two levels returns *"[I found some flights with availability in business class. None of them are on KLM.]*$^{1.level}$ *[[The only direct flight arrives at 5:30, which is later than you requested.]*$^{1.child}$ *[To arrive earlier, youll have to make a connection]*$^{2.child}$*]*$^{2.level}$*. [If youre willing to travel economy]*$^{1.level}$*, [[there is a direct flight on BMI, arriving at 1:10 pm.]*$^{1.child}$ *[Theres also a KLM flight arriving at 1:50, but it requires a connection in Amsterdam.]*$^{2.child}$*]*$^{2.level}$*"*. The next turn is then determined by the user choice which of the options to hear more about.

5.5 Attribute Selection

Section 5.3 described how content selection is done for the option level. This section concerns the choice of attributes that are mentioned to describe a flight. In the FLIGHTS system, the s-compellingness measure allows to choose attributes that describe why an option is compelling in comparison to some other option. The goal is to mention those features that are most relevant for decision finding.

For the modified system, the task stays the same. To effectively present tradeoffs, those attributes that constitute the tradeoffs need to be mentioned. Therefore, the justifications that were calculated in the pruning process determine the choice of attributes. The set of attributes to be mentioned for all options within a turn is determined as follows: In theory, all attributes that are in the justifications of some option (within a turn) get mentioned for all options in that turn. Practically, it is however not possible to mention each such attribute for every option, because there might be options for which the attribute is not homogeneous. If an attribute is not homogeneous for an option, it cannot be summarized effectively. Therefore, out of the set of attributes that are justifications, only the homogeneous ones get mentioned for each option. For example if one option is justified by the arrival time of its flights and the other is justified by the cheap prices of its flights, then the arrival times and prices of both options are mentioned, unless e.g. the first option is not homogeneous for prices, in which case only the arrival time

[2]generated based on query for user model "business" with preference for flights on KLM.

would be mentioned for the first option but the arrival time and price for the second option. There might then be some attributes, that are homogeneous for a node but that are not in the justification of any of the nodes. All these attributes constitute the first part of the following turn.

This strict rule for choosing attributes aims at keeping turns with many options concise and making them more coherent (because only a small fixed set of attributes is talked about). Turns that contain only three entities (which might occur further down the tree), are not object to the process described above. In such a case, the situation is similar to the one in the FLIGHTS system and complexity is not very high, so all attributes that exceed a certain relevance threshold are mentioned. The higher-level goal is to maintain a similar level of turn length for all turns. A more sophisticated method that also takes into account experimental data on understandability could be in the scope of future work.

In the FLIGHTS system, all attributes that are mentioned for the first flight are also mentioned about the second flight and vice versa. The information is “balanced”, in order not to leave the user with a huge question mark in her head (“Okay, you told me about the price of this flight... and how much is the other one?”). This cannot always be guarantied in this version of the system, because at a certain node in the cluster, only the homogeneous attributes are available. An example for a situation where an information cannot be given for all options would be a structure where there is one flight in the one cluster and four flights in the other. Then any information about the first cluster is available (because it must be homogeneous for all attributes), but saying everything equivalent about the second cluster would probably involve a lot of conjunctions of disjunctions, because some of the attributes might be heterogeneous across the four flights.
As there are several turns in the modified version of the system, the user can get her question answered later on in function of her choice and the problem does not apply.

Chapter 6

System Adaptation

This chapter describes briefly how the algorithms presented in chapter 5 were integrated into the content selection component and discusses the adaptation of the subsequent modules.

The most interesting issues came up in the adaptation of the content planner. Section 6.2 first points out the differences between the FLIGHTS content structure and the content structure needed for my version of the system. I then describe the content structuring strategy used in my system, explain how options are identified effectively, discuss alternative approaches to automatic summarization of attributes and finally examine the relationship between subsequent system turns.

The third section describes the adaptation of the interface component Plan2TextXml which translates the content plan to an xml tree which is then read in by the sentence planner.

Section 6.4 summarizes my work on the sentence planner. I explain how I adapted this module to generate the target logical forms and address the issues of generating references and packed alternative logical forms with XSLT.

The final two sections briefly present the realizer and the dialogue manager modules.

6.1 Integration of the Core Algorithm

The new code which implements the algorithms described in chapter 5 is mainly[1] contained in three new classes: ContentClusterer.java, EntityClusterer.java (which substitute the original classes ContentSelector.java and EntityEvaluator.java), and Cluster.java, which introduces a new class for the representation of clusters.
As the content selection step returns an option tree instead of an option list and contains some additional information (e.g. justifications), major changes to the method that converts the internal representation of the selected content into an xml tree had to be made. The xml tree is then transformed to a structure that can be read in by the LISP-based content planner component, using an XML stylesheet[2]. The same style-sheet is used for all turns. An example transformation is shown in figure 6.1. In addition to this modification, two new agent communication functions which allow to access the data for subsequent refinement turns (see 5.4) and methods to process them were added to the class OAAAgent.java.

A figure that shows the similarities and differences for the original FLIGHTS system user model output structure and the corresponding structure of the modified system is available in appendix D.

6.2 The OPLAN Content Planner

O-PLAN (Currie and Tate [1991]) is a schema-based planning language which was originally designed for artificial intelligence planning applications. It interacts with a LISP program which provides the functionality to test conditions for the use of a schema and retrieves the values which are needed to fill the slots of the content plan. This LISP component reads in the data structure produced by the content selection module, retrieves the necessary values from it and post-processes the data (e.g. to generate summaries, scalar terms).
The main task of the content plan component is to define what is mentioned when, what concepts should be contrasted and how things should be referenced.

[1] except one accessor method in the PartitionComponentFunction class
[2] The XSLT stylesheet is processed within the Java component using Java Xalan

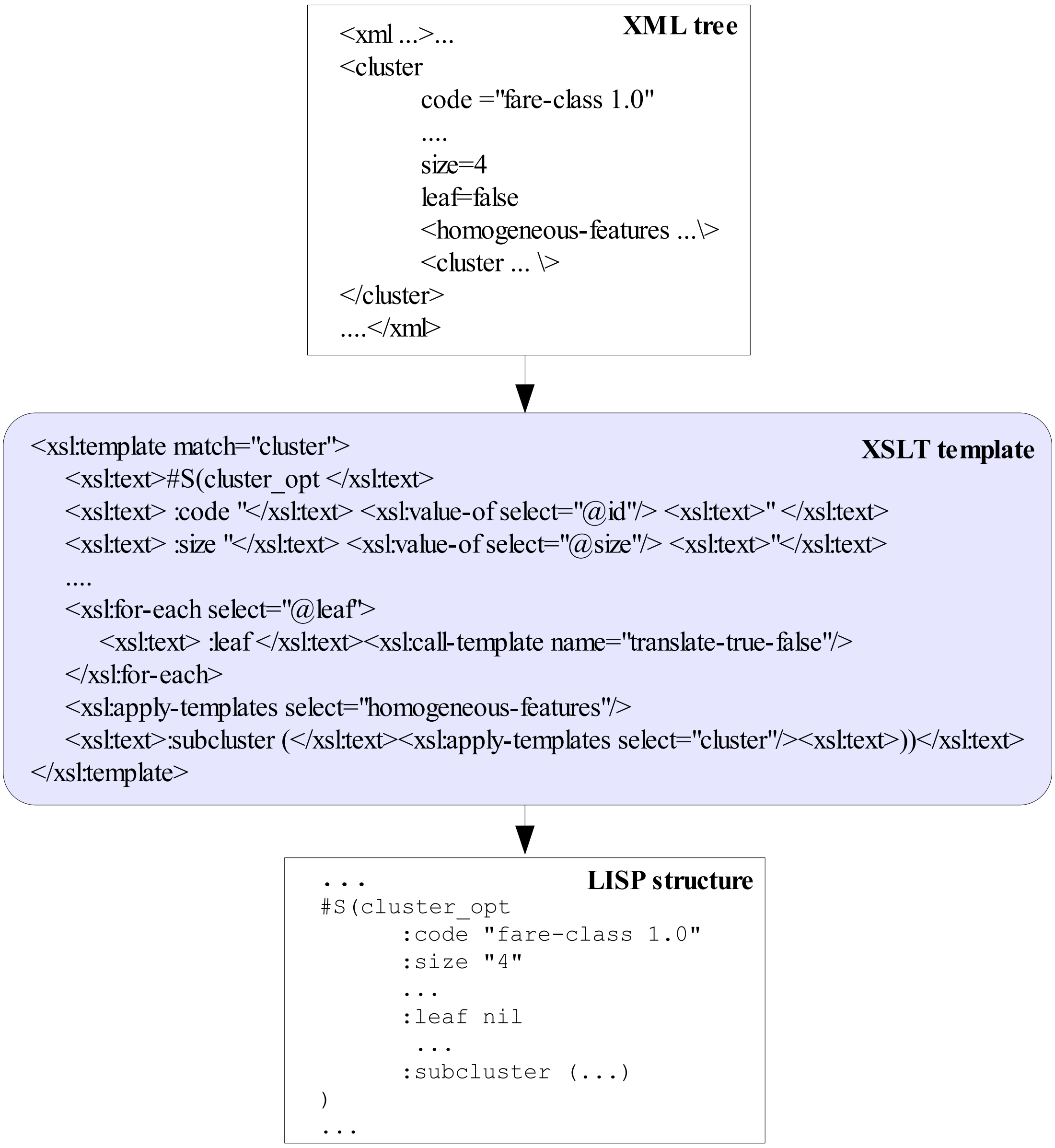

Figure 6.1: XSLT-Transformation from XML Tree to LISP Structure

6.2.1 FLIGHTS Content Plan Structure

In the original system, a content plan consists of a list of options, where each option is first identified and then elaborated. The elaboration consists of either a list of positive and non-negative attributes or of a contrast between a list of positive and other attributes and a list of negative attributes. This structure is made explicit through labels.
Each of the options is identified by the airline and either by its most attractive property or by the attribute that makes it compelling with respect to the previous option. The attribute lists are ordered: for lists of positive attributes from the strongest to the weakest argument, and for the list of negative attributes from the least relevant negative argument to the most relevant one, where relevance is dependent on the attribute value and weight.
The most negative attribute is always mentioned last for an option and becomes thus salient. As the compellingness-selection criterion picks options with compelling differences, the next option is likely be identified by its positive value for the salient attribute. Options that are identified by the contextually most salient attribute and make an easy to grasp contrast between the different options.
This idea was not fully implemented in the version of the FLIGHTS system at the time when I worked on it (salience of attributes was not checked).

6.2.2 Content Structure in my system

The input that the content planning component gets from the content selection module is quite different from the FLIGHTS content selection output. Instead of a list of flights, there is a limited-size option tree (because the pruned option tree is cut into chunks suitable for single turns). I considered to build up a content plan structure, where the options in the child node were embedded in the elaboration of an option.

The advantage of an embedded structure is its greater flexibility. It would enable both structures from figure 6.2, whereas a flat structure (for example appending the child-options after the main cluster) would make the second one impossible.
Dependent on strategies for content planning, there are two main reasons why one might want to mention an attribute that belongs to the parent node after the presentation of its children.

I found four flights	identification	I found four flights
with avail. in business class,	positive attribute	with avail. in business class.
but none are on KLM.	negative attribute	
The only direct flight	1. child identif.	The only direct flight
arrives at 8:30 pm.	1. child neg. attr.	arrives at 8:30pm.
To arrive earlier,	2. child identif.	To arrive earlier,
you'll have to connect flights	2. child neg. attr.	you'll have to connect flights.
	negative attribute	*None of the business class flights are on KLM.*

Figure 6.2: Alternative Content Structures

One strategy is to first mention all important positive attributes of an option set, followed by all important negative attributes of an option set, then talk about the properties of its children and finally mention those attributes that are homogeneous for a the option set but not very important for the current user. Less interesting attributes are usually very far down in the unpruned tree, but may get pushed up in the tree if they are homogeneous for the set of flights in consideration. Talking about the less interesting options later would have the advantage, that they are less forced into the focus of the user. This can prevent irritation about why they are mentioned, as they do not contribute very much to the decision but rather have the effect of additional information.
The second consideration is a bit more context-sensitive. When the next option is justified by an attribute for which the current option has a negative value, this negative attribute should be mentioned last (e.g. "... but it costs £1000. A cheaper flight..."). The attribute which will be used for the identification of the next option is then contextually salient, and a more coherent structure would be achieved.

A practical disadvantage of having an embedded structure was the necessity to adapt the content planner in laborious work. The original system used one sentence per option. With the embedded structure, such sentences would get much too long. An adaptation of the assignment of the sentence-tags would not have been a principal problem, but because of time limitations, I decided not do undertake this adaptation and keep to the one-sentence-per-option structure. I can see an interesting potential for future work here, that would investigate the benefit and naturalness of alternative structures such as proposed above.

The tree structure is flattened in first-in order (like in depth-first search), to make a list from the option tree. For content structuring, the same strategy as in the FLIGHTS system is used: options (sets of flights in this case) are first identified, then their good properties are listed, and contrasted against negative properties if there are any.

6.2.3 Identification of Options

Whereas options are identified by their highest ranked positive attribute in the original version of the system, the identification of an option set is based on its justification. A justification of an option set has the form of a disjunction of conjunctions of attributes (e.g. all flights in a set are either direct or arrive in time and are on business class; for an explanation, see section 5.3.3). If there is just one disjunct, identification is unproblematic: if one of the attributes in the conjunction is salient, this one is used. Otherwise the highest ranked attribute is taken.
If there is a disjunction of reasons (this can only happen if the option set has no justification of its own but is justified by its children, see 5.3.3), the option set cannot be identified by the highest ranked or salient justification, because this would summarize the set incorrectly. Instead, the option set is identified by its cluster criterion and marked as being non-positive. Such an identification can then be realized with a separate construction acknowledging that it's non-positive e.g. "If you're willing to travel economy / arrive later / accept a longer travel time, ...".
For the identification of sets, I added information about the set size, to enable constructions like: "There are four flights with availability in business class", which provides information for the correct use of singular and plural later on in the sentence planner.
Another difference to option identification in the original system is that for identification of flight sets, there is no special case for the airline attribute any more, as the airline is not homogeneous for every option set.

For every identification term of an option, it has to be decided if it is theme or rheme and definite or indefinite. The default is rheme and indefinite. "Theme" is used when the attribute by which the option is identified is contextually salient. Salience is determined by retrieving which attribute and value were mentioned last. For an example o-plan schema for theme identification see figure 6.3.

In the original system, "definite" is used when an option is identified by the attribute

```
schema do_identify_theme;
   vars   ?option_id = ?{satisfies atom},
          ?size = ?{satisfies atom},
          ?salient,
          ?summary =?{satisfies listp};
   expands {identify ?option_id};
   nodes  1 action {inform cluster ?option_id ?size status_given
                  definite_false info_theme},
          2 action {inform ?summary ?option_id};
   orderings 1 ---> 2;
   conditions only_use_if {time_mode} = VERBOSE,
          compute {get_size ?option_id} =?size,
          compute {get_salient ?option_id}=?salient,
          compute {null ?salient}=false,
          compute {find_ident_theme ?option_id ?salient} =?summary,
          compute {null ?summary}= false;
   effects {identified ?option_id} = true at end_of 2;
end_schema;
```

Figure 6.3: O-Plan Schema

that is globally most important for the user, e.g. "the cheapest flight" for the student user model. The new version uses "definite" for clusters, that are leaves of the cluster tree, i.e. for those options, where a cluster has already been mentioned before and is mentioned again to give more detailed information. The target sentences that were created before-hand also contained some definites "the only direct flight" or "the two flights with shortest travel time". An investigation of natural flight information dialogues would be needed to find a rule for the use of such definite phrases rather than "There is one direct flight" or "There are two flights with a short travel-time". I did neither have time nor resources to do this, so I left this feature underspecified and let the sentence planner generate both a definite and an indefinite version and choose (based on n-grams) a formulation on the basis of a small corpus containing our target sentences.

6.2.4 Summarization of Attributes

In the elaboration part of the content plan, all positive homogeneous attributes are mentioned and contrasted against all average or negative attributes. An attribute that was used for identification of an option is not mentioned again in the elaboration. In opposition to a single flight, attributes may have different values for the entities within a set of flights. In that case, these attribute values need to be summarized. There are three main cases to be distinguished:

1. The continuous values for the attributes "price" and "arrival/departure/travel-time" need to be summarized, as they may differ in their values even if they are in the same cluster. There are two ways to summarize them. The first is to use an expression that reflects their value range, e.g. "between (x y)". A problem I found was that texts using those expressions seemed very unnatural. Consider for example: "There are two flights with a travel time *between 4h 20 min and 4h 30 min*. They 're both on British Airways and cost 308 pounds."
   There are several possible explanations for this:

   - the small cluster size - it might be more acceptable for clusters of larger size
   - the exact range - smoothing might improve naturalness, for example "There are four flights arriving between 4 and 5 pm."
   - not tailored enough - determine the relevant property of the values, for example "before 5 pm" or "after 4 pm"

   I chose another solution, which is to only mention the evaluation value, leading to sentences like "The two flights *with shortest travel time* are both on British Airways and cost 308 pounds." or "The *cheapest* flights."

2. For the discrete-valued attributes with a small number of possible values "number-of-legs" or "fare-class" summarization is not an issue, because when homogeneous for a cluster, attribute values are identical for all flights of the cluster.

3. The third group of attributes are those like "airline" and "layover-airport". The summarization for these is based on the work of Polifroni *et al.* [2003], who use expressions like "Most of them are located in Boston and Cambridge." or "None of them are expensive." For the flights domain, fuzzy expressions like "most of them", "many of them" or "predominantly" are inappropriate, because the clusters are smaller and it is insatisfactory to hear "I have found 5 flights, they are predominantly on BMI.". So for the flights domain, the only quantifications used are "all of them", "both of them", "none of them" and "either of them". However, if there are many different values, this would end up in formulations like: "They are on RyanAir, BMI, British Airways and Lufthansa.". Singleton-out, as

> proposed by Polifroni for such cases, does not work either, because there are still several possible airlines which those flights are not on.
> At this point, the user model can come back into play for a tailored summarization, that makes use of the user preferences (for example liking KLM and disliking Ryan Air). Then, the above example could be summarized as "None are on KLM.", which is shorter and responds to the user's interest.

To be able to do summarizations that include negations, it was necessary to add further information to the data. In its original version, preferences were only stored as "reasons" on the attribute level of each option. Each flight-object knew whether it was on a preferred airline or not. So when seeing a flight with the airline BMI, it would be possible to know, by the lack of the reason attribute, that this airline was not preferred by the user. But it was not possible to retrieve the preferred attribute in order to say "not X". To fix this problem, the preferences are now read out from the user model data and stored globally. The preference values (which are only available in their encoded form, e.g. KL instead of KLM Airlines, LH instead of Lufthansa or LHR for London/Heathrow) need to be decoded. For this purpose, a dictionary was was built up that used the data-base in which both the code of an airline or airport and its natural language description are available (for an example data base entry see appendix B). Preferences in the form of a list for liked values and disliked values were then stored in the data structure that is given to the oplan component. These preference lists can not only contain categorical preferences but also situational user-requests like the desired arrival time, which enable formulations of the following type: "*To arrive earlier*, you would have to make a connection.". An issue arising from summarization with negation is that the negated value has to be salient, otherwise the utterance might be irritating. For a user that dislikes London/Heathrow as a layover-airport, it would be better to say "These flights are not direct." in a neutral context, but "You would not need to connect in London Heathrow." in case a salient option connected there.

Finally, all attributes that had to be summarized, need to be mentioned again later on when the cluster has been narrowed down to one entity. Otherwise the user would have a question in mind: "Okay, I know you say the flight has a good price, but how much is it exactly?" or "The airline is not KLM, but what is it?" Therefore, for each branch in the tree, the summarized attributes are stored and their concrete values mentioned once a leaf of the tree is reached (after subsequent turns). The correct set of attributes that need to be mentioned again is retrieved from a list that is keyed on the paths through the tree.

6.2.5 Subsequent Turns

After having listened to a turn, the user should have the possibility to refine her query. Therefore, she needs to identify the branch of the tree she wants to hear more about. The content planning component sends this request to the content selection component, which constructed the whole option tree and stores the chunks into which that tree has been chopped. The content selector needs a sequence of cluster codes (e.g. “arrival-time good”, “price average”) to retrieve the following turn. I hypothesized, that the most natural way to refer to the options is via the identification terms, that were used by the system. (But this was disproved in evaluation, see section 8.3.7.) The content planner therefore stores the mapping from cluster identifications to cluster codes.

For this project, the system was only run with text in- and output instead of speech recognition and speech synthesis. The content planner provides an access to the possible choices and in a human readable form.
“Please choose which cluster you want to hear more about.
1) business class and direct
2) business class and arrival time good
3) economy class and travel-time short
4) economy class and KLM”

For the integration of speech synthesis and speech recognition, no such choice would be presented to the user. Instead, the user would just ask for one of the options they heard about. This would necessitate to recognize a more flexible set of references to options, so that all attributes that were used for a cluster work as identifiers.
In the evaluation of this project, participants were asked in one of the questions to give a free text answer in which they had to identify the best option they heard about. Investigation of this data will show whether my hypothesis was correct and give an indication, of how future users will refer to options and can therefore be used for constructing a robust recognition for refinement queries. For results see section 8.3.7.

6.2.6 Technical problems

The use of many different programming languages requires complex interfaces between the components. These interfaces ensure that each component gets its input in a form that it can process and so that no reserved words or symbols are used. A technical problem I encountered was that I used the format "02:35" for time attribute values (for arrival time, travel time etc.), which contains a colon. This was no problem for building up the content plan, I got a fine output for the html content plan and was very surprised to discover, that the method get_hier_plan that's in the standard OPLAN package always failed saying that it could not find "package 02". The solution, which I found with the help of Claus Zinn, was that the function used the lisp method read_from_string(), which tries to evaluate all strings as symbols. So the colon in times as "02:35" was not protected within the string. The format of times was therefore changed to 02_35, which could easily be done in the XSLT style-sheet that translates the user model XML tree to LISP notation.

The planning language OPLAN used for the content planner is described in Art [1997b] Art [1997a].

6.3 Plan2TextXml - the content planner - sentence planner interface

The Plan2TextXml module is the interface between the content plan component and the sentence planner. It reads in the content plan in the form in which it is returned by the OPLAN get_hier_plan method (see appendix E) and transforms it into XML format, which can then be processed by the sentence planner and a simple text format, which is quite easily understandable for humans, even though it's not in full sentences. For example:

The input to the component is decomposed on the basis of its bracket structure. The only adaptation I had to make was to enable the processing of the additional summarization structures that were needed for the content plan of clusters. These additional structures comprise: or($x_i...x_n$), and($x_i...x_n$), between(x_1, x_2) and not(x). The unary structure not(x) is flattened so that an attribute negation="true" is simply added to the

```
cluster business-class available airline not KLM.
cluster direct arrival-time 20:30.
cluster arrival-time eval earlier indirect.
cluster economy-class available arrival-time between(16:10,16:50).
cluster direct airline BMI.
cluster airline KLM connection Amsterdam.
```

original structure. The binary structure between(x_1, x_2) is also flattened and three attributes are added to the structure: summary="between", val1=x_1 and val2=x_2 instead of the attribute "val" in which a single value would be stored. For the n-ary structure of or(x_i...x_n) and and(x_i...x_n), one new node for each argument was inserted into the structure.

Another function of the plan2textxml component is to translate some structure terms, for cases where the distinction between different terms introduced by the content planner is not needed in the sentence planner. For example, I had to use some other labels for the sequences and turns planning steps for turns that are not the first turn, which can be written back to the same name in this component. Also, the format for times is changed back from 02_35 to 02:35 and some values are rewritten, so that they can be processed by the realizer, e.g. klm_airlines is rewritten to KLM.

6.4 The XSLT Sentence Planner

The sentence planning component uses XSLT to transform the content plan tree into logical forms for sentences. The sentence planning agent is implemented in Java and uses Apache Xalan to process the XSLT templates. XSLT was chosen for sentence planning in the FLIGHTS project because it combines the traditional template-based approach to sentence planning with AI-planning and is well suited for the kind of tree-to-tree transformations needed here (from content plan trees to logical-form trees). The use of XSLT for the FLIGHTS project is described in Foster and White [2004].

XSLT templates match content plans with particular properties. I introduced e.g. different templates for singular and plural sentences and for options which are identified by a positive vs. a negative property. A template produces a logical form for the realizer. To do this, it may call other templates and insert their output into its own structure to generate the final result.

My work on this component consisted of extending and newly introducing several high-level templates in order to produce sentence structures that were not needed in the FLIGHTS system, such as identifications by negative properties (e.g. "If you're willing to [make a connection]$_{vp}$"[3]) or templates for plural sentences (e.g. "There are four flights ..., all of them [are direct]$_{vp}$.").

I also had to add some templates to generate logical forms for summaries. There are two ways of summarizing times and prices: either with a "between" or "and"-construction or by scalar terms, such as "good", "earlier", "faster" (refer to 6.2.4). For an example of how this can be handled in XSLT, see figure 6.4 which shows, for VPs, the distinction between cases where a scalar term is used and cases, where the concrete value or a "between"-summarization is applied.

```
<xsl:template match="inform[@pred='arrival-time' and
                            varmap:get($mymap,'mode')='vp']">
  <node id="{idgen:next($mygen)}" pred="arrive">
    <rel name="Theme">
      <node id="{idgen:next($mygen)}"/>
    </rel>
    <xsl:choose>
      <!-- arrive earlier / later -->
      <xsl:when test="@eval">
        <rel name="TimeRel">
          <node id="{idgen:next($mygen)}" pred="{@eval}"
                kon="+"/>
        </rel>
      </xsl:when>
      <!-- arrive at 12 pm / between 3 pm and 4 pm -->
      <xsl:when test="not (@eval)">
        <xsl:call-template name="sum"/>
      </xsl:when>
    </xsl:choose>
  </node>
</xsl:template>
```

only applied if a VP is needed

calls another template that can handle summaries as well as single values

Figure 6.4: XSLT template for Sentence Planning

Lexical choice is performed through the selection of logical form templates. Often, there are alternative formulations for the same content. For example [connect flights]$_{vp}$ would be an equally fine formulation as [make a connection]$_{vp}$. The sentence planner has no basis to decide which formulation is better, so all alternative structures are passed on to the realizer in a packed representation.

[3]Logical forms for constituents in square brackets are filled in by other templates

I shortly want to explain how references are dealt with in this framework. Sometimes, long references have to be made to allow for the use of correct pronomina or other long dependencies. Consider for example: “You’d need to connect in Manchester”, where “you” is the object of both “need” and “connect”. To maximize flexibility, “connect in Manchester” would be processed by another template, and the higher-level “You’d need to X”-template would require some vp, so that it could be re-used for “You’d need to fly economy class.”
The required phrase-mode (vp / np / s etc.) and the referent are stored in a global hash-map from which they can be retrieved by the matching template.
XSLT also easily handles aggregation by matching several bits of information with the same template and combining them to one sentence.

I implemented all sentence structures needed for the Walter1 target dialogue (see appendix G.2), but did not have the time to implement all sentence structure for all target dialogues. This is one reason, why the evaluation (see chapter 7) could not be done with participants using the system in a real scenario.

6.5 The OpenCCG Realizer

The realizer was kindly adapted for my target sentences by Michael White, providing me thereby with the output specification for the sentence planner. The input to the realizer module consists of logical forms with embedded alternatives. The OpenCCG-realizer is based on a version of Combinatory Categorial Grammar (CCG) that can also process theme/rheme information and determine the type of pitch accents and the type and placement of boundary tones, producing APML[4]-annotated text.
The sentence planner outputs alternative logical forms in a packed representation. The realizer uses a corpus of target sentences to statistically determine the best realization on the basis of n-grams. So the final choice of the surface form is not done by the sentence planner but by the realizer. The realizer handles issues that would be difficult to implement in XSLT, such as syntactic agreement.
A helpful manual for OpenCCG in the form used in this and the FLIGHTS system is Bozsahin *et al.* [2005].

[4] Affective Presentation Markup Language

6.6 The Dipper Dialogue Manager

The Dipper dialogue manager directs the dialogue flow. It has an internal information state and chooses the next action for the dialogue based on conditions on its information states. Answers that come back from the agents are stored in variables within the information state and may then constitute the input for the query to the next agent. To adapt the dialogue manager to my system, I extended the dialogue script so that the choice is presented to the user in the text input/output window. The user can then type in her choice, which is sent to the appropriate agents and thus puts the final piece into place to make more than one user-system turn possible.
For a description of Dipper see Bos *et al.* [2003].

Chapter 7

Evaluation

A within-participants laboratory experiment was conducted in order to determine whether user model-based clustering leads to increased overall user satisfaction, a better overview of the available options and quicker accessibility to the optimal option. It furthermore assessed whether the options were presented in a way that was well understandable.

Each subject was presented with six dialogue pairs, where each dialgue was followed by five questions and each pair by one further forced choice question. Each dialogue pair consisted of one dialogue between a user and my system and one dialogue between a user and a system designed as proposed by Polifroni *et al.* [2003] and Chung [2004]. The order of the dialogues within a dialogue pair was randomized. The dialogues were provided as transcripts.

The first section of this chapter motivates the experiment. I then give some technical data about the participants, the experiment procedure and address issues concerning the experimental design. Finally, I state the measures (questions and measurements) used in evaluation and provide a brief characterization of the five dialogue pairs.

The results of the evaluation, which are very promising, are discussed and analysed in detail in chapter 8.

7.1 Experiment Purpose

The experiment served to evaluate the system presented here against the Polifroni system Polifroni *et al.* [2003] and Chung [2004] with respect to understandability, quality of overview of available options, user confidence in having heard all relevant options, ease to find the optimal flight for a request, overall user satisfaction and dialogue duration. Furthermore it provided a way of collecting data of natural human references to options. Care was taken to design the dialogues that represented the Polifroni system according to the design descriptions for the restaurant recommendation database described in Polifroni *et al.* [2003] and Chung [2004]. System answers which can not yet be fully generated by my system (because of incompleteness of the sentence planner templates) were based on their content plan. Also, the goal was to make the dialogues sound as natural as possible.

7.2 Participants

A total of 38 adults participated in the experiment. Email notification of the experiment was sent to the informatics student mailing list. Individual participants interested in participation contacted the experimenter and were assigned a slot on one of two days on which the experiment was conducted. All of the participants were university students or PhDs, so the group was not balanced with respect to age and education. There were both native and non-native speakers of English among the participants. 22 out of the 38 participants were females and 16 males. All participants had the same conditions, except for the order of dialogues within a dialogue pair; this order was randomized. Participants were paid £5 for participation. It took them between 20 and 50 minutes to complete the experiment. All participants gave informed consent.

7.3 Experimental Design

The experiment was designed using E-Prime Schneider *et al.* [2002]. E-Prime allows the exact measurement of response times and collects all the data (measured response times as well as answers) into a table. It allows to merge data from different subjects into one big table and facilitates data analysis by allowing an easy access to the relevant data entries in the table and allowing for plots of basic analyses (mean, variance, counts etc.) by dumping the data directly into excel. It also rewrites the data into a form that can then be read into SPSS to allow for more complex analysis.

The experiment was first run with four pilot subject to refine and improve the experimental procedure and check questions and instructions on clarity.

All participants were presented with exactly the same conditions using the same computers, so that differences in equipment across subjects should not influence results. Up to two subjects did the experiment at the same time, sitting at different machines in the same room. There was a small wall between them so they could not see each other or each other's screens. Subject were not allowed to talk to one another during the experiment.
All participants who wanted to take part in the experiment were first handed a consent form which informed them about their rights and of what to expect. Participants indicated their agreement by consenting to take part in the experiment and were then lead to the lab in which the experiment took place. The first screen of the experiment informed them of their tasks in the experiment and contained instructions. Participants proceeded from screen to screen in their own time.

The experiment consisted of 12 dialogues. These dialogues were grouped to six dialogue pairs, each containing one dialogue generated by my system and one dialogue, that the Polifroni system would have generated for the flights domain, according to Polifroni *et al.* [2003] and Chung [2004]. Each dialogue pair was based on the same data base and the same request from a user. The example user was designed according to one of the three user-models contained in the FLIGHTS system. These user models are easy to distinguish and memorize, in that they try to capture prototypical types of users. Choosing these user models should minimize cognitive effort for the subjects that needed to remember the user models and judge the dialogues according to them. There were two dialogue pairs for each the frequent flyer ("Carol"), the business client ("Walter") and the student ("Steve"), each of whom had one request for a flight from Edinburgh to Brussels and one request for a flight from Edinburgh to San Francisco.
The order of dialogues within a dialogue pair was randomized to relativate the influence of ordering. The ordering can have several different effects that risk to bias the results: Firstly, when asked to answer the forced choice question, participants might

remember the dialogue they heard last better than the first one and therefore more often choose the system from the second dialogue. Secondly, as both system answers were based on the same data base and request, knowing about the response of the other system might influence participants' judgment of the system in the Likert scale questions. Furthermore, reading times can be expected to be shorter for the second dialogue, as participants already know about the available options.
The first dialogue pair was presented to the participants as a training run and was not evaluated.

In this experiment, dialogues were presented to the subjects in the form of transcripts. Alternatively, a user reading the requests to the system could have been recorded and the system answers could have been synthesized and recorded. An advantage of using transcriptions is that no influencing intonation (for example in the form of hesitations or an intonation that judges the system responses (sounding happy / confused etc.)) is introduced into the dialogues. In this respect, text is the most neutral form of presentation.
A disadvantage is that the situation for reading is different from the one for listening. Humans have a better short time memory for information they read than for information they heard. Also, it is not possible to go back in spoken information. So the fact that users are able to process a certain amount of information when reading it, does not allow to conclude that they can process the same amount of information when it is presented to them as speech. To minimize these effects of textual information presentation, dialogues were shown on a screen one turn at a time. When they had finished reading, subjects were asked to press the space bar to proceed to the next turn. The time they took to read the turns was recorded. This time recording allows to see whether much more time was spent on longer turns than on shorter ones. If the correlation between turn length and reading time is not linear, this suggests that participants needed to re-read the information.
Another way of determining if information complexity was much higher for one system than for the other was recording whether the users wanted to reread a dialogue: after each dialogue, participants were given the choice whether they wanted to read the whole dialogue once again. Rereading the dialogues should also allow the participants to make up their minds for the questions following the dialogues.

7.4 Measures

Each dialogue pair was followed by the forced choice question "Which of the two systems would you recommend to your friend?". This question aimed to assess with which of the systems the subject was more satisfied. The answer "neither" was not allowed.
After each dialogue (i.e. twice in a dialogue pair) there was one question that served the purpose of giving the subjects the task to concentrate on the options, in order to put them into the position of a person that has a request for a flight and seeks an answer rather than a simple spectator. "Which option do you think was best for X[1]?" . Subjects were allowed a free text answer of up to 200 characters. The answer to this question is furthermore used to analyse how participants refer to options, in particular if they use the same identification term as the system.
Then there were four questions that asked the user to express their opinion about the dialogue system responses according to different aspects. The second, third and forth questions were based on seven-point Likert scales and the fifth question on a three-point Likert scale.

1. Which option do you think was best for X[1]?
   free text answer - max. 200 characters
2. Did the system give the information in a way that was easy to understand?
   1: very hard to understand
   7: very easy to understand
3. Did the system give you a good overview of the available options?
   1: very poor overview
   7: very good overview
4. Do you think there may be flights that are better options for X[1] that the system did not tell X[1] about?
   1: I think that is very possible
   7: I feel the system gave a good overview of all options that are relevant for X[1].
5. How quickly did the system allow X[1] to find the optimal flight?
   1: slowly
   3: quickly

[1] where X was instantiated by Steve, Walter or Carol

Participants responded by typing the answer on the keyboard (response time was not measured). When participants finished with all questions on one dialogue, they went on to the next dialogue.

Dialogues were chosen carefully to focus on different aspects of the travel agents. All of the Demberg dialogues were designed based on the content plans which the system generates (because the sentence planner does not yet cover all constructions). The Polifroni dialogues were generated based on the same data base entries, following the descriptions in Polifroni *et al.* [2003] and Chung [2004]. Here is a brief characterization of the dialogues (the full dialogues are provided in appendix G.1 – G.5):

1. dialogue (Carol1)
   - backtracking required in Polifroni dialogue
   - short and comprehensible in Demberg system (just two flights mentioned)
2. dialogue (Walter1)
   - two long system utterances in Demberg dialogues
   - too restricted first query in Polifroni dialogue
3. dialogue (Carol2)
   - optimal flight found directly in Polifroni system
   - Demberg system mentions flight that is not mentioned by Polifroni
   - Demberg dialogue might seem incoherent because not fully balanced
4. dialogue (Steve2)
   - very long system utterance in Demberg system (all attributes mentioned in one turn)
   - Polifroni system mentions flight that is not mentioned in Demberg dialogue
5. dialogue (Walter2)
   - rather uninformative summary for a set of 7 flights in Polifroni system
   - three flights mentioned in detail

Chapter 8

Results

The experiment described in the previous chapter yielded very promising results. I will first introduce the statistical methods I used to test for significance of the results obtained.

In the following, I discuss the evaluation results for the forced choice question, each of the Likert scale questions concerning understandability, accessibility of relevant options, quality of overview and confidence in having heard about all relevant options. I then analyse the data collected from the reading-time data. Finally, I summarize how people referred to flights when answering the free choice question.

8.1 Statistical Methods

8.1.1 The T-Test

The t-test tests the null hypothesis, that the means of two normally distributed populations are equal. If the variances are also assumed to be equal, the “Student’s t-test” can be applied, otherwise a similar test, called Welch’s t-test can be used.
There are two variants of the Student’s t-test: an *independent* t-test is conducted when the two samples are independent from one another. A *paired* t-test is chosen when “each member of the one sample has a unique relationship with a particular member of the other sample.” (citing http://en.wikipedia.org/wiki/T-test).
Lastly, one has to decide whether to use a *one-tailed* or *two-tailed* t-test. A one-tailed

test should be chosen, when there is some basis to predict the direction of the difference (e.g., if they differ at all, then A can only be smaller than B). Otherwise, a two-tailed t-test is used (the two-tailed test is "more strict").
For the questions that used a Likert scale, a two-tailed paired Student's t-test was used: For these questions, the null-hypothesis is that the mean scores are the same for both dialogues in a dialogue pair and variances are assumed to be equal. Furthermore, each score for a Demberg dialogue has a unique corresponding score for the Polifroni dialogue (the one that was given by the same person for the same question).

I calculated the t-values using the t-test function available in MS Excel. The null hypotheses that the two groups do not differ is rejected if the calculated t-value is greater than a certain critical value. The critical value for statistical significance is usually set to 0.05. If the value for t is below 0.05, the probability that the two distributions differ by chance is $< 5\%$.

8.1.2 The Binomial Test

"The binomial test is an exact test of the statistical significance of deviations from a theoretically expected distribution of observations into two categories."
(from http://en.wikipedia.org/wiki/Binomial_test)
The binomial test applies to the data obtained in the forced choice question, where subjects had to answer either '1' or '2' to indicate which system they preferred. The expected distribution of observations (the null-hypothesis) is again, that both systems are preferred equally often.
I used an online - binomial test calculator[1] to asses the significance for the force choice questions.

8.2 Overview of Results

A significant preference of the Demberg system was observed. There were a total of 190 forced choices in the experiment (38 participants * 5 dialogue pairs). The Demberg system was preferred 120 times (0.6316%), the Polifroni system was preferred only 70 times (0.3684%)(see figure 8.1). This difference is highly significant (p-value

[1] http://www.graphpad.com/quickcalcs/binomial1.cfm

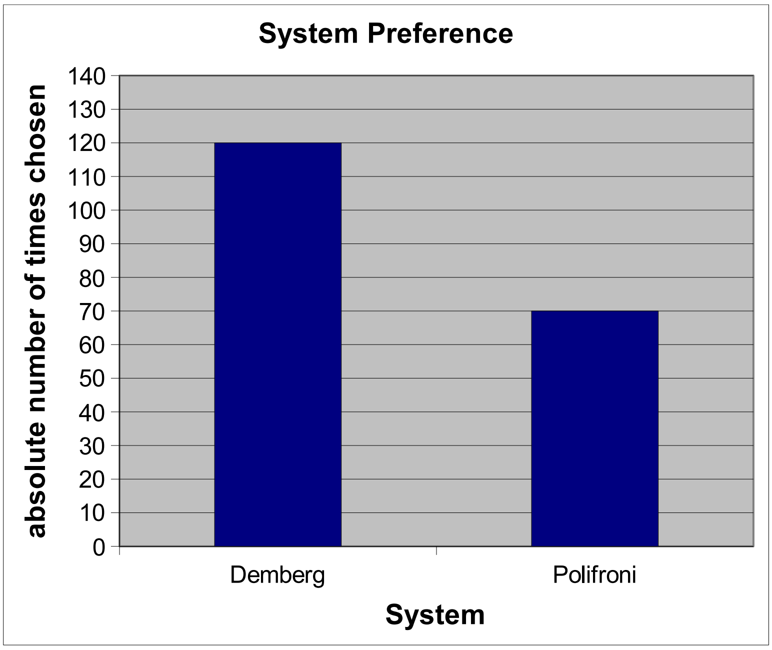

Figure 8.1: Result for Forced Choice Question

= 0.0004) according to a two-tailed binomial test. This means that the probability of observing a more extreme difference (less than 70 votes for the Polifroni system and more than 120 for the Demberg system) by chance is 0.0004. Thus, the null-hypothesis, that both systems are preferred equally often, can be rejected with high confidence. Although this result seems excellent, it has to be treated with care. The more in-depth analysis shows that there was a significant bias towards the dialogue heard last and a considerable variance across dialogue pairs.

The evaluation results for the Likert scale questions correspond to what I expected. The Polifroni dialogues got on average slightly higher scores for understandability (question 2), which can be explained by the shorter length of the system turns for that system. However, the difference is very small and not statistically significant according to a two-tailed paired t-test. For a difference in distribution to be considered significant, the p-value must be <0.05. The p-value for the answers to question 2 is 0.97, this means that the probability that the observed data is from the same distribution is 97%. The differences in results for the other questions are highly statistically significant, especially for question 3 (which assessed the quality of overview of the options given by the system responses), where the p-value is <0.0001 and question 4 (assessing the confidence that there all relevant options were mentioned by the system), which has a p-value of $7<0.0001$. The difference for question 5 (accessibility of the optimal flight) is statistically significant at a level of p=0.0003. The average scores (based on 190 values) are shown in figure 8.2. (NB: the range for answers to question 5 is not based on a scale from 1 to 7 but from 1 to 3.)

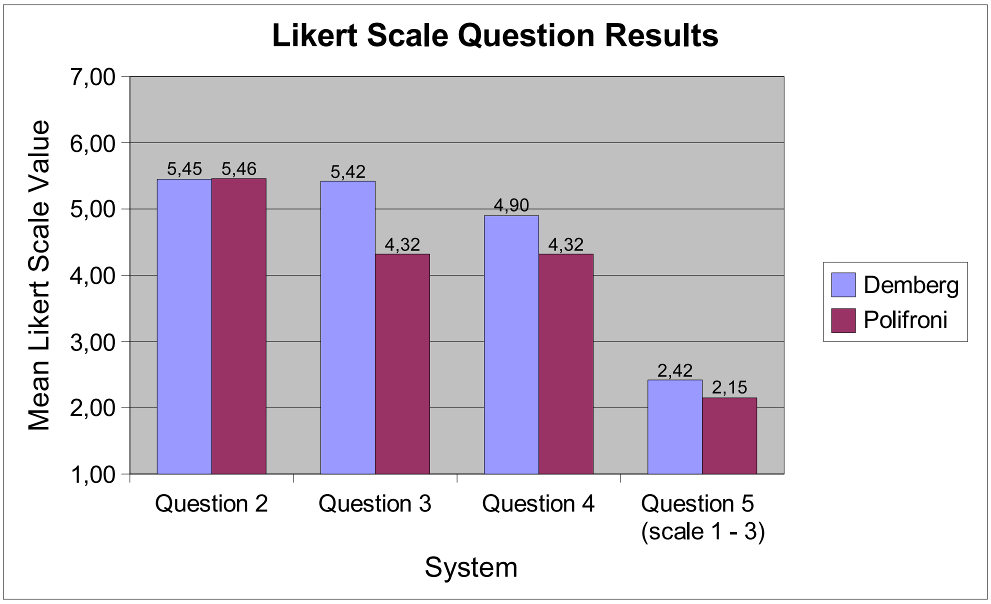

Figure 8.2: Results for all Questions

8.3 Detailed Discussion

8.3.1 System Preference

A closer look at the data showed that participants choice is significantly biased (p=0.0243) towards the system they had listened to most recently. The order of dialogues within a dialogue pair was randomized, so that this bias should not make the results invalid. Figure 8.3 shows the distribution for the two positions in the dialogue pair. Despite the bias, the choice seems consistent. The p-value for the first dialogue is 0.0066 and 0.0223 for the second dialogue. The rate of choosing my system is higher if the Polifroni dialogue was heard first (76% vs. 62%). But this difference is not significant, i.e., the null-hypothesis that these two rates come from the same distribution cannot be rejected.

Dialogues were chosen such that they show different aspects of the two systems. Fig-

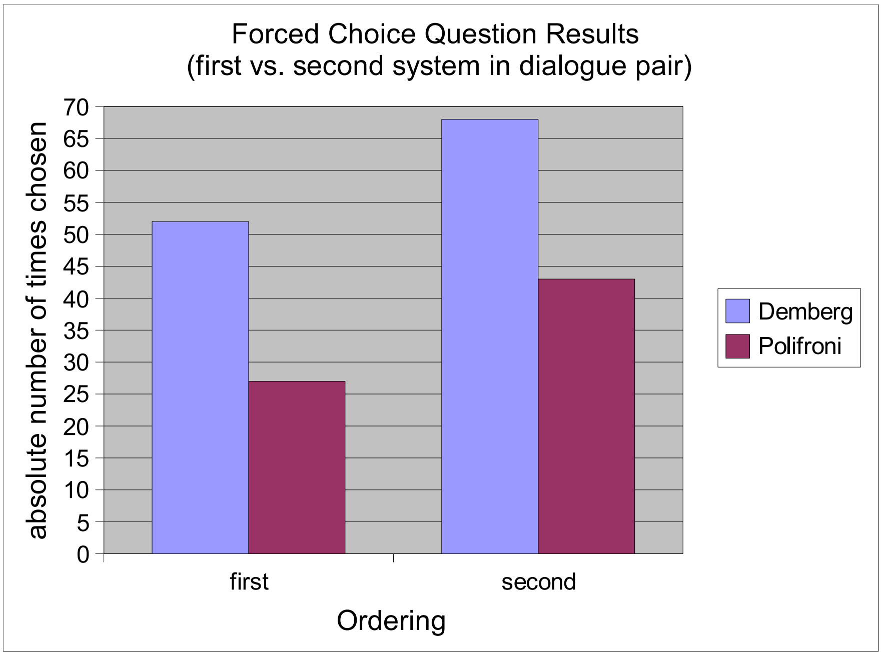

Figure 8.3: Bias towards most recently heard Dialogue

ure 8.4 shows the subject preferences broken down into single dialogues.

In general, for queries with few potentially conflicting aspects (e.g., no restrictions on airports or airlines and no strict restrictions on arrival time), the probability that some near to optimal option exists is high. In such a case it is less essential to hear about all possible tradeoffs. I hypothesized that in such a case the shorter and less complex response from the Polifroni system would obtain a higher score for understandability and accessibility of the optimal flight. The users would not value being informed of alternative tradeoffs and be more satisfied with the Polifroni system for this dialogue. The student user model is such an example: the only aspect the user cares about is the price (see appendix G.4).

The first business user model dialogue (Walter1), involves particularly long turns in my system, so I hypothesized that the Polifroni dialogue would be preferred for understandability (see figure 8.5) and task duration(see figures 8.8 and 8.9).

The data confirms these hypotheses, even though the differences are not large enough to reach significance level: For the second dialogue, the probability of observing this choice distribution by chance is 25% and for the forth dialogue this probability is 62%. On the other hand, for dialogues 1, 3 and 5, my system was chosen significantly more often. For the first dialogue with the frequent flier, Carol, the difference is highly significant at $p < 0.0001$. For the third dialogue, the chance of observing this or a more extreme distribution is <0.02 and the p-value for dialogue 5 also reaches significance level ($p<0.002$).

This considerable dependence on particular dialogues suggests that the systems have complementary strength and weaknesses, which become visible in different data and query situations. To make general statements about the systems rather than their performance in different situations, both approaches would need to be implemented in the same application domain and many more dialogues would have to be generated and evaluated to make more valid statements about the "average" dialogue. A detailed

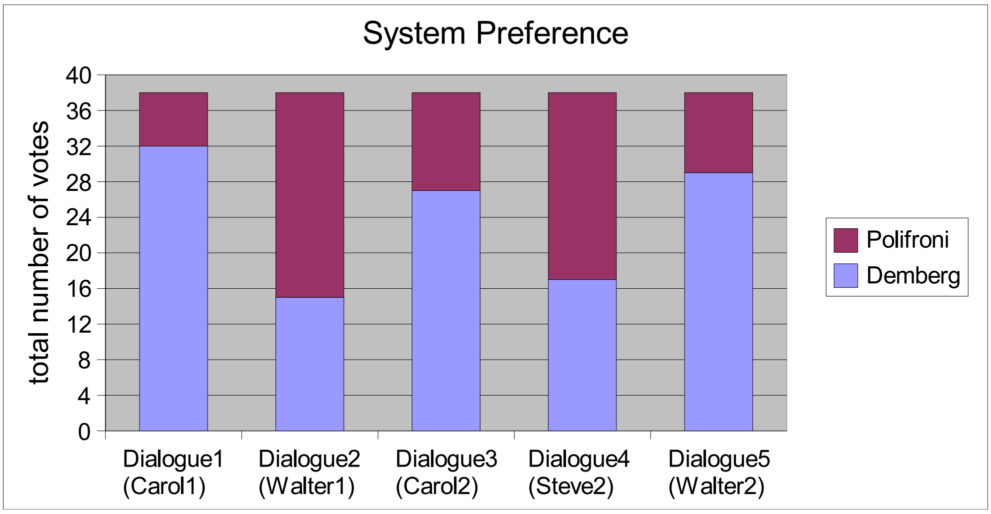

Figure 8.4: User Satisfaction with each of the Dialogues

analysis of the factors that cause participants to choose one rather than the other system will be discussed in the next section, based on the data from evaluations of the questions that followed each dialogue.

8.3.2 Understandability

The results shown in figure 8.5 were obtained by asking the question "Did the system give you the information in a way that was easy to understand?". Subjects answered this question by giving a score on a scale from 1 (=very hard to understand) to 7 (=very easy to understand). As the above overview showed, the differences in mean scores for understandability were not significant for the overall distribution. When broken down on single dialogues however, differences in performance for each dialogue pair could be observed. I had expected the participants to report that the Polifroni system was more understandable, because the amount of information per turn was generally smaller than in the corresponding Demberg dialogues.

In two of the dialogue pairs, the Demberg dialogue got a higher average score, the remaining three dialogue pairs had a higher score for the Polifroni dialogues. All differences but the one for dialogue 2 were significant at $p<0.02$.

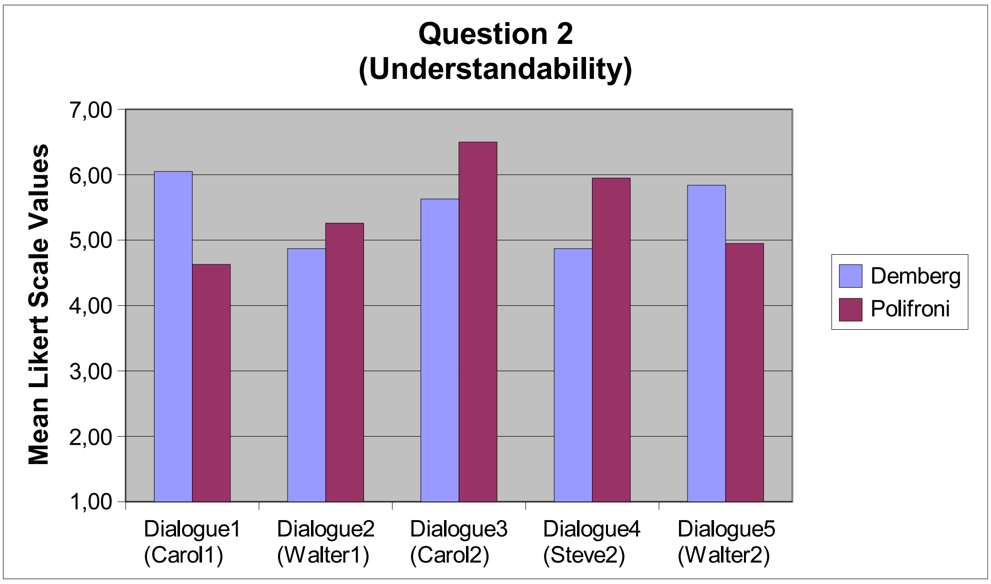

Figure 8.5: Results for Question 2

Dialogues 1 and 5 have two common properties that can explain the higher scores for the Demberg system. Both dialogues have a more complex structure for the Polifroni system than for the Demberg system. In both dialogues the initial user query is too restricted, such that no flight in the data base can satisfy it. Whereas in the Demberg dialogue, this situation leads to the presentation of alternative tradeoffs, the user in the Polifroni dialogue is informed that no flight satisfies the query. To support the user more actively in her search for a good option, one of the constraints is relaxed to make an alternative suggestion. Following the constraint relaxation strategy of Chung [2004], the constraints that produce the smallest set of options are maintained. Constraint relaxation optimizes the subset size rather than attribute relevance. When the flights mentioned happen to be uninteresting, the user needs to make a new query with different restrictions (for an example see the second interaction from Polifroni dialogue 1 in appendix G.1 or second interaction from Polifroni dialogue 5 in appendix G.5).

When there is a non-empty set of options that satisfy the query, one additional attribute is mentioned about this set of options. This additional information aims at keeping the dialogue fluid and quick (the more information is given, the less turns are necessary) while at the same time minimizing memory load on the user (by mentioning just one attribute rather than several). Another problem that might have led the subjects to score the Polifroni system lower arises from the strategy of choosing this additional attribute. It is not determined based on what is important to the user (due to the lack of a user model) but by optimal summarizability, i.e. the attribute with the most homogeneous values is chosen. If this attributes happens to be one that does not contribute much to the decision process, it could be perceived as irritating and hence not easy to understand.

I hypothesize, that the need to make a new query (i.e., explore different branches of the option tree) and the mentioning of a not very relevant attribute caused the subjects to report lower understandability for the Polifroni dialogue. Furthermore, the low number of only two tradeoffs to be presented in the Demberg system in dialogue 1 probably caused this dialogue to have the highest understandability scores within the Demberg dialogues.

8.3.3 Goodness of Overview

The design of my system has a strong focus on giving the user a good overview of the options by mentioning alternative tradeoffs and summarizing irrelevant flights. The Polifroni system only mentions alternatives in the case where no entries could be found that match the request. Due to the lack of a user model, the presentation of an alternative might not focus on the aspects which are relevant to the user.

When there are no data entries that match the user's request, the user has to try another request in order to find the optimal tradeoff. The system does not support the user in finding this tradeoff. The user then has to try different branches of the tree and therefore might get lost and not feel confident about having found the globally best option for herself. In my system, by summarizing two tree levels in one turn, a "preview" of the alternative tradeoffs is provided and the description focuses on the most relevant aspects of the options. Bad options are summarized so that the user should feel confident, she did not miss out an option.

Therefore, I expect the user to give a higher score to my system on quality of overview and confidence that all relevant options were presented.

Figure 8.6 shows the results for the question “Did the system give a good overview of the available options?”, using a Likert scale from 1 (=very poor overview) to 7 (=very good overview). This question tries to assess coherence within a turn (because an incoherent system turn would not be described as providing a good overview) and coherence of the whole dialogue (here a dialogue where several branches need to be explored is less coherent). Another factor that comes into answering this question is how complete participants felt the information was. This was asked in a separate question (see 8.3.4).

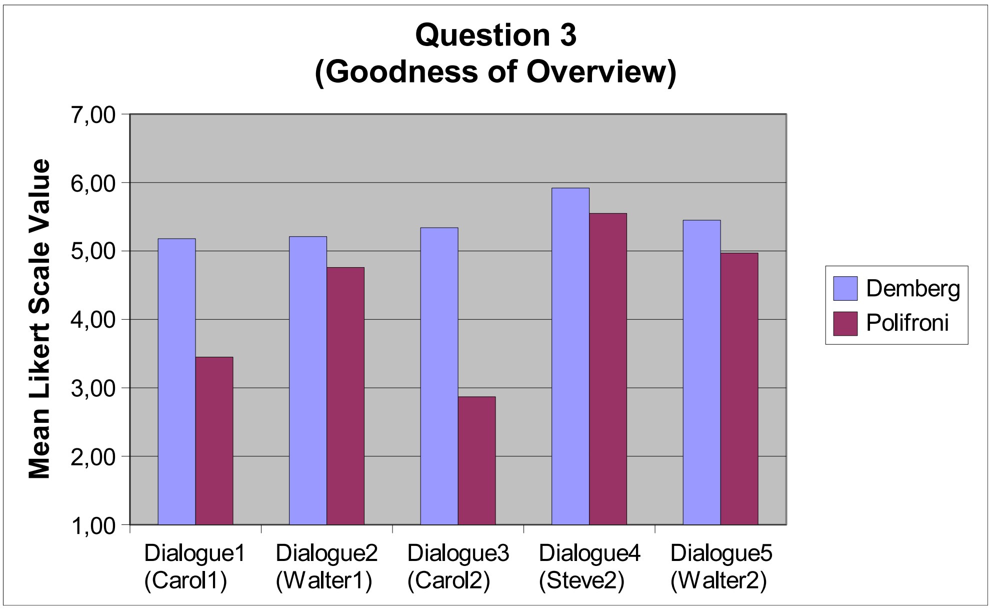

Figure 8.6: Results for Question 3

As shown in figure 8.6, the Demberg system was consistently rated higher than the Polifroni strategy, receiving higher average scores for every dialogue. The difference is highly significant (at $p < 1E^{-5}$) for dialogues 1 and 3, but does not reach significance level for the other dialogues. For dialogue 1, the large difference can be explained by the need for the user to ask a new question in the Polifroni dialogue (as explained in 8.3.2), but this is also the case for dialogue 5. Another factor that can account for the large difference in dialogue 1 is the fact that the second user query had just one restriction, which was satisfied by 7 flights. This set it too large to go into any detail about these flights (the Demberg system would have pruned out dominated flights and have mentioned the most important property about the remaining flights), and may therefore have caused participants to feel they did not have a good overview of the available

options.
The low score for dialogue 3 can be explained differently. In this dialogue, only one flight satisfied the initial user query. In such a case, it is the responsibility of the user to ask other queries to find out about other flights. At no point does the Polifroni system provide a summary about flights in the data base that do not satisfy the query. The Demberg system presented three alternatives for that query, even though one was clearly best (the one that was mentioned in the Polifroni dialogue). The other two options featured properties that were not of great importance to the user. Furthermore, a summary about the remaining data base entries was provided.

8.3.4 Confidence in having heard all relevant information

This data was collected by asking the question "Do you think there may be flights that are better options for X that the system did not tell X about?" (where X was instantiated with the name of the user in that dialogue). The scale given was from 1 (=I think that is very possible) to 7 (=I feel the system gave a good overview of all options that are relevant for X).
For dialogue 3 the difference in mean scores is highly significant (see figure 8.7). The high difference in scores between systems can be explained by the fact that just one option was mentioned and no summary of the other data entries was given by the Polifroni system. In another case, dialogue 4, the Polifroni dialogue mentioned a flight that was not mentioned by the Demberg system, which presumably caused the Polifroni system to have a higher score in this question. But this difference is much smaller and does not reach significance level. This phenomenon can be explained by the fact that the Demberg system provides summaries for dominated flights, so that subjects still felt confident that they did not miss out relevant information.
For the first dialogue, both systems have an average score that is lower than the medium scale value, which means that participants felt very unsure about flights that might be interesting but were not mentioned. The only difference that I think could explain this effect is that for both dialogues in dialogue pair 1, only two flights are described in detail[2], whereas for all other dialogues (except the Polifroni dialogue in dialogue pair 3), at least 3 options are described in detail. Further investigation would be necessary to find further evidence for this theory.

[2]By "described in detail" I mean mentioning of at least two attributes.

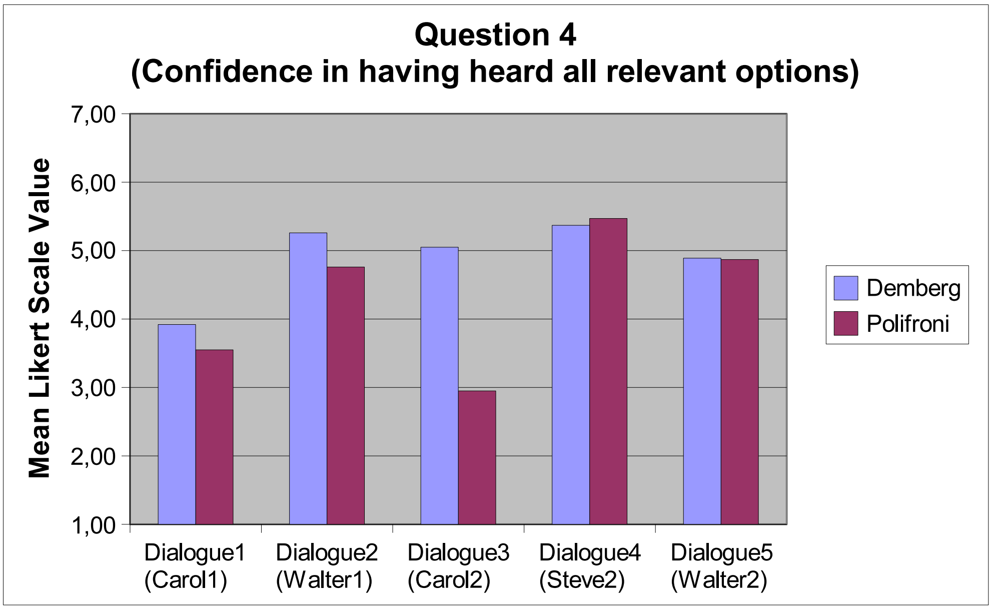

Figure 8.7: Results for Question 4

Questions 3 and 4 assess very similar concepts. In order to asses the strength of the correlation between these two distributions, Pearson's test for correlation was conducted between the results of these two questions. This test measures the strength of the linear relationship between two variables. For the Polifroni dialogues, a strong positive correlation was found (Pearson correlation coefficient of 0.78), and for the results of the Demberg dialogues, another factor seems to play a greater role in assessing the quality of overview, but a significant positive correlation was still found (Pearson correlation coefficient of 0.54). Importantly, the fact that a correlation was observed, cannot show any causal relation between the two distributions but rather suggests, that they assessed a similar question. A correlation coefficient of -0.3 to +0.3 is generally interpreted as "little or no association", a correlation coefficient of +0.3 to +0.7 can be interpreted as a positive correlation and a correlation coefficient above +0.7 as a strong positive correlation[3] For comparison, the Pearson correlation coefficient between the results from questions 2 and 3, which I do not expect to correlate is only 0.24 for the Polifroni dialogues.

[3]For a more detailed significance level table see http://www.une.edu.au/WebStat/unit_materials/c6_common_statistical_tests/test_signif_pearson.html

8.3.5 Accessibility of target options

Participants could answer question 5 “How quickly did the system allow X to find the optimal flight?” (where X was instantiated with the name of the user in that dialogue) with scores on a scale from 1 (= slowly) to 3 (=quickly). Results are shown in figure 8.8.

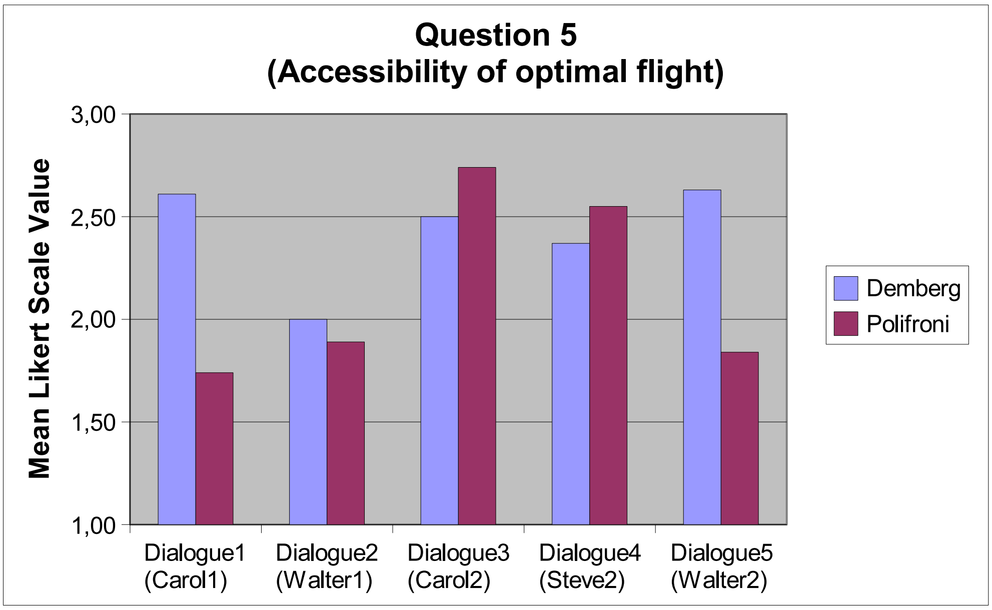

Figure 8.8: Results for Question 5

In dialogues 1, 2 and 5, the user explored several tree branches when communicating with the Polifroni system. This is the most likely reason for the highly significant (at $p< 0.0001$) differences in scores between the systems in dialogues 1 and 5. Interestingly, for dialogue 2, where the difference is not significant, backtracking was required at a low level (i.e., when the overall direction was good) and seems to be less negative than backtracking at an early level. A weakness of the Polifroni system that shows here is that the determination of usefulness of an attribute is based on a criterion for efficient summarization instead of a user model.

Dialogues pairs 3 and 4 have better scores for the Polifroni version of the system, but they are not statistically significant. In these dialogues, the initial query happened to directly retrieve a very small subset of flights. The optimal flight was mentioned by both systems in the second system turn, but it was perceived to be found quicker when less other flights were mentioned in the same turn.

I had expected a linear correlation between average total dialogue duration and perceived quickness to find the optimal flight. For four out of the five dialogues, average reading times were usually longer for the Demberg system (see figure 8.9), but this did not reflect in the results for question 5.

I therefore suggest that dialogue duration be optimized with respect to perceived quickness to access an option, instead of objective dialogue length. An investigation on the correlation between each of the two measures and user satisfaction should be conducted in future work.

For a detailed discussion of the measured reading times see section 8.3.6.

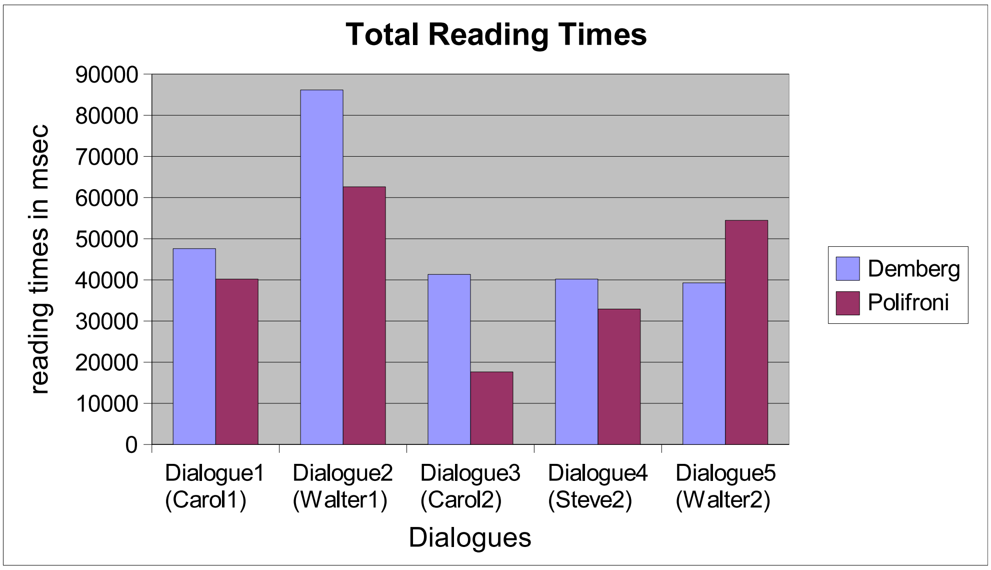

Figure 8.9: Average Total Dialogue Reading Times

8.3.6 Reading Times

The times participants took to read a dialogue turn were logged, in order to investigate whether those reading times were linear or not. Participants were allowed to read the dialogue turns in their own time and were not told that their reading times were logged, so that they would not try to speed up. A big difference between written information and information that is presented orally is that participants might go back in written text and read complex information again. If reading times do not increase linearly with respect to turn length, this means that the content is too complex and would be very hard to follow when presented orally to users. The dialogue system would then not be suited for communication via mobile phone or similar. For these dialogues however, the reading times were found to increase linearly with the number of characters (see figure 8.10).

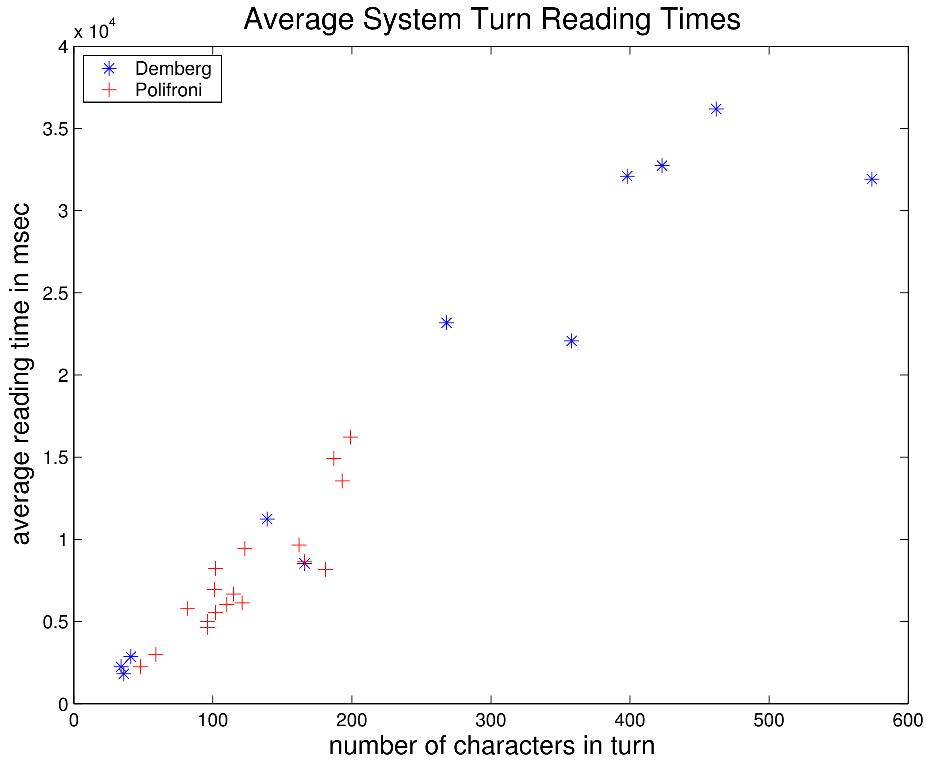

Figure 8.10: Correlation between Reading Times and Turn Length

After listening to a dialogue, participants were given the chance to read the dialogue once again. The dialogue that was read again least often was the Demberg dialogue from dialogue pair 1, which was just reread 6 times, and only 9 subjects chose to read the Demberg dialogue from dialogue pair 4 again. This indicates that these dialogues were straightforward, easy to memorize and coherent. All other dialogues were reread by 13 or 14 participants, except for the Demberg dialogue from dialogue pair 3, which was reread by 20 out of the 38 participants.
Dialogue 3 is short, but slightly incoherent, because (for reasons of clustering) the arrival time of two of the flights is not mentioned in the first turn but only in the refinement turn. This incoherence may have confused or irritated participants and caused them to read the dialogue again.
For all dialogues, participants who read the dialogue again gave on average slightly lower scores in question 2 (that asked for understandability). This effect could be ob-

served throughout all dialogues and a two-tailed t-test (not paired, as the number of times participants reread a dialogue was lower than the number of times they did not read the dialogue again.) confirms that the difference in average scores for question 2 is significant at $p < 0.02$.

8.3.7 Referencing Options

The flights were referred to in very diverse ways. Most people preferred to use the airline for identification, and did not always pick up the identification terms used by the system.

11% of the answers contained descriptions that were not contained word-by-word in the system description of the flight ("the flight before 3", "the flight near noon" or "the short flight") or referred to the order the flights were mentioned in ("the second KLM flight", "the first flight").

6% of the answers were overspecificied (when more attributes than necessary were mentioned) and 13% of the answers were underspecified (which is not specific enough to identify a single flight). The underspecification cases can perhaps be explained by the task, which asked the user to say which flight would be best for Steve, rather than to book one flight. Some participants were undecided and wrote e.g. "one of the RyanAir flights".

These results suggest that humans use very diverse terms for identifying the options they heard about. Simply keying on the identification terms used by the system as I did in this first implementation is not sufficient. Unsurprisingly, a keyword spotter that recognizes not only the references from the identification but also terms used in the elaboration of an option, would not be robust enough either. Instead, reasoning would be needed to interpret answers like "the flight near noon".

Chapter 9

Conclusion

9.1 Conclusion

I investigated the synthesis of two earlier approaches to generation in spoken dialogue systems. This thesis extends the approach proposed in Moore *et al.* [2004] by not only identifying relevant options but by also structuring them for gradual refinement. It thereby incorporates methods presented in Polifroni *et al.* [2003] that allow for effective summarization of option sets and increased flexibility with respect to the data base by clustering attribute values. Data presentation is driven by the user model, the actual dialogue context and the available data. The system generates user-tailored information that is designed such that it actively supports the user in making decisions between multiple options involving relevant complex tradeoffs.

I demonstrated how the methods proposed can be integrated into an existing dialogue system and discussed a wide range of issues that come up in this concrete context. Among these were the automatic generation of coherent text, design decisions that increase user confidence in the system, determination of a compromise between conciseness and correct presentation of tradeoffs, and various issues concerning the optimal summarization of option sets and their effective presentation.

A comparative evaluation between the proposed system and a system designed following the descriptions in Polifroni *et al.* [2003] and Chung [2004] has shown that user satisfaction could be increased. In particular, subjects reported that the system provided them with a better overview of the available options and that they felt more certain to have been presented with all relevant options. Results suggested that even though the presentation of complex tradeoffs usually requires relatively long system turns, participants could still cope with the amount of information presented. Although total dialogue duration was longer in my system than in the other system, this did not have a negative effect on user satisfaction. Subjects even reported for some of the dialogues that they felt they could access relevant options quicker despite longer overall duration.

9.2 Future Work

9.2.1 Evaluation under Real Conditions

An evaluation with subjects using the system in "real" conditions, e.g., in order to complete a task (for example booking a flight to some destination on a certain airline) would be desirable. One reason why this was not possible in the scope of this project was that I focused on the design and implementation of the clustering algorithm and did not have enough time to adapt the speech recognition component and the speech synthesis component, and did not finish the sentence planner component. Furthermore, although I extended the sentence planner and realizer to generate several of the utterances used in the evaluation, I could not guarantee that the system could generate all necessary sentences in a real interaction. A further obstacle was insufficient robustness and the fact that the system would have to be optimized, such that it runs in real time.

9.2.2 Information Complexity

For the complex information my system presented, participants took a longer time to read the dialogue than for the equivalent amount of information presented by the Polifronian version of the system. Even though participants did not report that my system was harder to understand on average, it would be interesting to see how well users can understand and remember information from the system when a part of their concentration is absorbed by another task, for example when using the system while driving a car.

9.2.3 Evaluation of the Clustering Algorithm

A possible evaluation of the clustering algorithm would be to compare how often subjects try to investigate options from different tree branches in a version where the clustering algorithm was based on their specific user model for clustering vs. in a version where it was based on a user model that is sufficiently different from their own one. Having to investigate different branches of the tree is an indicator that the tree structure was not optimal. In an optimal tree, the user should at every point be able to choose the optimal branch directly. To do this evaluation, the sentence planner would have to be extended to cover all the desired output sentence structures. For this form of evaluation, text input and output would be sufficient, so the speech recognition and speech synthesis components would not have to be adapted. Robustness would still be an issue, but subjects could be instructed to only use a fixed syntax that is not problematic for processing, or input would be based on templates, as already mentioned above. The risk that this manipulation would have a biasing effect on the result of this particular evaluation seems minor to me, because the only parameter that changes in this evaluation is the user model that underlies the tree structure. The input form should not have any impact on that.

9.2.4 Determination of Optimal Conciseness

I hypothesize that user satisfaction with the system is dependent on the conciseness of the cluster summary, which is controlled by the content selection component (see 5.5 and 5.3). The influence of conciseness on the performance of the system can be estimated by measuring user satisfaction while the only aspect changed in the system is conciseness. This kind of evaluation was successfully done in several recent systems (Carenini and Moore [2004], Walker *et al.* [2002]). Parameters to be optimized are the number of attributes that are mentioned for an option, the number of options in a turn and the level of detail with which bad options are summarized. At the level of summarizing bad options, two conflicting aspects need to be optimized: accounting for all available options in detail increases, so I hypothesize, user confidence. On the other hand, long summaries about irrelevant options make the dialogue inefficient. Confidence can be bought by paying with efficiency. It will be interesting to investigate, what the ideal balance is for this pair.

Appendix A

User Model “Business Traveller”


```
<?xml version="1.0" encoding="UTF-8"?>
<UserModel instantiated="true">
    <Rankings>
        <Group rank="1">
            <Objective ref="fare-class"/>
        </Group>
        <Group rank="2">
            <Objective ref="arrival-time"/>
            <Objective ref="departure-time"/>
            <Objective ref="number-of-legs"/>
            <Objective ref="travel-time"/>
        </Group>
        <Group rank="6">
            <Objective ref="airline"/>
        </Group>
        <Group rank="7">
            <Objective ref="layover-airport"/>
            <Objective ref="price"/>
        </Group>
    </Rankings>
    <ComplexObjective name="Overall evaluation of flight"
        description="" queryVars="">
        <PrimitiveObjective name="fare-class" weight="
            0.16405895691609979" queryVars="">
            <PartitionComponentFunction variable="classes">
              <Name value="business" eval="Like"/>
              <Name value="economy" eval="Don't Care"/>
              <Value type="Dislike" value="0.2"/>
              <Value type="Don't Care" value="0.5"/>
              <Value type="Like" value="0.8"/>
            </PartitionComponentFunction>
        </PrimitiveObjective>
```

```
        <PrimitiveObjective name="number-of-legs" weight="0.16405895691609979" queryVars="">
            <ComponentFunction variable="legs">[...]</ComponentFunction>
        </PrimitiveObjective>
        <PrimitiveObjective name="layover-airport" weight="0.032312925170068035" queryVars="">
            <PartitionComponentFunction variable="layover_airport_0_1">
                <Name value="MAN" eval="Don't Care"/>
                <Name value="LHR" eval="Dislike"/>
                ...
                <Name value="CDG" eval="Don't Care"/>
                <Value type="Dislike" value="0.2"/>
                <Value type="Don't Care" value="0.5"/>
                <Value type="Like" value="0.8"/>
            </PartitionComponentFunction>
        </PrimitiveObjective>
        <PrimitiveObjective name="airline" weight="0.3704081632653061" queryVars="">
            <PartitionComponentFunction variable="airline_code_0">
              <Name value="BA" eval="Don't Care"/>
              <Name value="KL" eval="Like"/>
              ...
              <Name value="FR" eval="Dislike"/>
              <Value type="Dislike" value="0.2"/>
              <Value type="Don't Care" value="0.5"/>
              <Value type="Like" value="0.8"/>
            </PartitionComponentFunction>
        </PrimitiveObjective>
        <PrimitiveObjective name="arrival-time" weight="0.16" queryVars="desired_arrival_time">
            <ComponentFunction variable="arrival_time">[...]</ComponentFunction>
        </PrimitiveObjective>
        <PrimitiveObjective name="travel-time" weight="0.072" queryVars="">
            <ComponentFunction variable="travel_time">[...]</ComponentFunction>
        </PrimitiveObjective>
        <PrimitiveObjective name="price" weight="0.072" queryVars="">
            <ComponentFunction variable="price">[...]</ComponentFunction>
        </PrimitiveObjective>
    </ComplexObjective>
</UserModel>
```

Appendix B

Data Base Entry


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Entity-List>
    <Entity name="KL2176/KL691">
        <Attribute key="airline_code_0" value="KL"/>
        <Attribute key="airline_code_1" value="KL"/>
        <Attribute key="airline_name_0" value="KLM Airlines"/>
        <Attribute key="airline_name_1" value="KLM Airlines"/>
        <Attribute key="airplane_type_0" value="Boeing 757 200"/>
        <Attribute key="airplane_type_1" value="Boeing 737-500 Jet"/>
        <Attribute key="arrival_date" value="2003.06.27"/>
        <Attribute key="arrival_date_0" value="2003.06.27"/>
        <Attribute key="arrival_date_1" value="2003.06.27"/>
        <Attribute key="arrival_day_0" value="Friday"/>
        <Attribute key="arrival_day_1" value="Friday"/>
        <Attribute key="arrival_time" value="16:20"/>
        <Attribute key="arrival_time_0" value="08:45"/>
        <Attribute key="arrival_time_1" value="16:20"/>
        <Attribute key="arrival_time_of_day" value="1500 - 1800"/>
        <Attribute key="classes" value="[business, economy]"/>
        <Attribute key="classes_0" value="[business, economy]"/>
        <Attribute key="classes_1" value="[business, economy]"/>
```

```
        <Attribute key="departure_date" value="2003.06.27
           "/>
        <Attribute key="departure_date_0" value="
           2003.06.27"/>
        <Attribute key="departure_date_1" value="
           2003.06.27"/>
        <Attribute key="departure_time" value="06:00"/>
        <Attribute key="departure_time_0" value="06:00"/>
        <Attribute key="departure_time_1" value="14:10"/>
        <Attribute key="departure_time_of_day" value="
           0600 - 0900"/>
        <Attribute key="flight_number_0" value="2176"/>
        <Attribute key="flight_number_1" value="691"/>
        <Attribute key="from_airport_0" value="EDI"/>
        <Attribute key="from_airport_1" value="AMS"/>
        <Attribute key="from_city_0" value="Edinburgh, UK
           "/>
        <Attribute key="from_city_1" value="Amsterdam,
           Holland"/>
        <Attribute key="layover_airport_0_1" value="AMS"/
           >
        <Attribute key="layover_city_0_1" value="
           Amsterdam, Holland"/>
        <Attribute key="layover_time_0_1" value="325"/>
        <Attribute key="legs" value="2"/>
        <Attribute key="meals_0" value="Meal Served"/>
        <Attribute key="meals_1" value="Snack/Brunch"/>
        <Attribute key="to_airport_0" value="AMS"/>
        <Attribute key="to_airport_1" value="YYZ"/>
        <Attribute key="to_city_0" value="Amsterdam,
           Holland"/>
        <Attribute key="to_city_1" value="SanFrancisco,
           USA"/>
        <Attribute key="from_airport" value="EDI"/>
        <Attribute key="to_airport" value="YYZ"/>
        <Attribute key="economy_price" value="348.00GBP"/
           >
        <Attribute key="business_price" value="1140.90GBP
           "/>
    </Entity>
    <Entity name="KL2176/KL691">
      ...
     </Entity>
</Entity-List>
```

Appendix C

Option Tree

C.1 Unpruned Tree

```
fare-class 1.0: 16 [BD649, BD627, BA1660/BA1618, BA4039/BA4976,
    LH6652/SN2174, BA4033/BA4974, BA1455/BA404, BE279/BA4958, LH4507/
    LH4436, BA1770/BA4964, LH6515/LH6703, LH6517/LH6705, BA1445/BA398
    , BA1443/BA396, LH6681/LH6707, BD61/BA404]
    airline 0.0: 16 [BD649, BD627, BA1660/BA1618, BA4039/BA4976,
        LH6652/SN2174, BA4033/BA4974, BA1455/BA404, BE279/BA4958,
        LH4507/LH4436, BA1770/BA4964, LH6515/LH6703, LH6517/LH6705,
        BA1445/BA398, BA1443/BA396, LH6681/LH6707, BD61/BA404]
        number-of-legs 1.0: 2 [BD649, BD627]
            layover-airport 1.0: 2 [BD649, BD627]
                price 1.0: 2 [BD649, BD627]
                    travel-time 1.0: 2 [BD649, BD627]
                        arrival-time 0.0: 1 [BD649]
                        arrival-time -1.0: 1 [BD627]
        number-of-legs 0.0: 14 [BA1660/BA1618, BA4039/BA4976, LH6652/
            SN2174, BA4033/BA4974, BA1455/BA404, BE279/BA4958, LH4507/
            LH4436, BA1770/BA4964, LH6515/LH6703, LH6517/LH6705, BA1445
            /BA398, BA1443/BA396, LH6681/LH6707, BD61/BA404]
            travel-time 1.0: 6 [BA1660/BA1618, BA4039/BA4976, LH6652/
                SN2174, BA4033/BA4974, BA1455/BA404, BE279/BA4958]
                arrival-time 1.0: 2 [BA1660/BA1618, BA4039/BA4976]
                    layover-airport 0.0: 2 [BA1660/BA1618, BA4039/BA4976]
                        price 0.0: 2 [BA1660/BA1618, BA4039/BA4976]
                arrival-time 0.0: 3 [LH6652/SN2174, BA4033/BA4974,
                    BA1455/BA404]
                    layover-airport 0.0: 2 [LH6652/SN2174, BA4033/BA4974]
                        price 1.0: 1 [LH6652/SN2174]
                        price 0.0: 1 [BA4033/BA4974]
                    layover-airport -1.0: 1 [BA1455/BA404]
                        price 0.0: 1 [BA1455/BA404]
                arrival-time -1.0: 1 [BE279/BA4958]
                    layover-airport 0.0: 1 [BE279/BA4958]
                        price 0.0: 1 [BE279/BA4958]
            travel-time 0.0: 8 [LH4507/LH4436, BA1770/BA4964, LH6515/
                LH6703, LH6517/LH6705, BA1445/BA398, BA1443/BA396,
                LH6681/LH6707, BD61/BA404]
```

```
            arrival-time 1.0: 6 [LH4507/LH4436, BA1770/BA4964,
                LH6515/LH6703, LH6517/LH6705, BA1445/BA398, BA1443/
                BA396]
                layover-airport 0.0: 2 [LH4507/LH4436, BA1770/BA4964]
                    price 1.0: 1 [LH4507/LH4436]
                    price 0.0: 1 [BA1770/BA4964]
                layover-airport -1.0: 4 [LH6515/LH6703, LH6517/LH6705
                    , BA1445/BA398, BA1443/BA396]
                    price 1.0: 2 [LH6515/LH6703, LH6517/LH6705]
                    price 0.0: 2 [BA1445/BA398, BA1443/BA396]
            arrival-time 0.0: 2 [LH6681/LH6707, BD61/BA404]
                layover-airport -1.0: 2 [LH6681/LH6707, BD61/BA404]
                    price 1.0: 1 [LH6681/LH6707]
                    price 0.0: 1 [BD61/BA404]
fare-class 0.0: 12 [BD629, KL1282/KL1731, CB766/VG156, BE6766/VG156,
      FR815/FR46, KL1286/KL1735, CB782/VG154, BE6782/VG154, FR813/FR44
    , BE6724/VG158, FR817/FR48, KL1276/KL1725]
    number-of-legs 1.0: 1 [BD629]
        airline 0.0: 1 [BD629]
            layover-airport 1.0: 1 [BD629]
                price 1.0: 1 [BD629]
                    arrival-time 1.0: 1 [BD629]
                        travel-time 1.0: 1 [BD629]
    number-of-legs 0.0: 11 [KL1282/KL1731, CB766/VG156, BE6766/VG156,
          FR815/FR46, KL1286/KL1735, CB782/VG154, BE6782/VG154, FR813/
        FR44, BE6724/VG158, FR817/FR48, KL1276/KL1725]
        layover-airport 0.0: 11 [KL1282/KL1731, CB766/VG156, BE6766/
            VG156, FR815/FR46, KL1286/KL1735, CB782/VG154, BE6782/VG154
            , FR813/FR44, BE6724/VG158, FR817/FR48, KL1276/KL1725]
            arrival-time 1.0: 4 [KL1282/KL1731, CB766/VG156, BE6766/
                VG156, FR815/FR46]
                travel-time 1.0: 3 [KL1282/KL1731, CB766/VG156, BE6766/
                    VG156]
                    airline 1.0: 1 [KL1282/KL1731]
                        price 1.0: 1 [KL1282/KL1731]
                    airline 0.0: 2 [CB766/VG156, BE6766/VG156]
                        price -1.0: 2 [CB766/VG156, BE6766/VG156]
                travel-time -1.0: 1 [FR815/FR46]
                    airline -1.0: 1 [FR815/FR46]
                        price 1.0: 1 [FR815/FR46]
            arrival-time 0.0: 6 [KL1286/KL1735, CB782/VG154, BE6782/
                VG154, FR813/FR44, BE6724/VG158, FR817/FR48]
                travel-time 1.0: 4 [KL1286/KL1735, CB782/VG154, BE6782/
                    VG154, FR813/FR44]
                    airline 1.0: 1 [KL1286/KL1735]
                        price 1.0: 1 [KL1286/KL1735]
                    airline 0.0: 2 [CB782/VG154, BE6782/VG154]
                        price -1.0: 2 [CB782/VG154, BE6782/VG154]
                    airline -1.0: 1 [FR813/FR44]
                        price 1.0: 1 [FR813/FR44]
                travel-time 0.0: 2 [BE6724/VG158, FR817/FR48]
                    airline 0.0: 1 [BE6724/VG158]
                        price 0.0: 1 [BE6724/VG158]
                    airline -1.0: 1 [FR817/FR48]
                        price 1.0: 1 [FR817/FR48]
            arrival-time -1.0: 1 [KL1276/KL1725]
```

```
        airline 1.0: 1 [KL1276/KL1725]
          price 1.0: 1 [KL1276/KL1725]
            travel-time 0.0: 1 [KL1276/KL1725]
```

C.2 Pruned Tree

```
fare-class 1.0: 4 [BD649, BA1660/BA1618, BA4039/BA4976, LH4507/
  LH4436]
  airline 0.0: 4 [BD649, BA1660/BA1618, BA4039/BA4976, LH4507/
    LH4436]
    number-of-legs 1.0: 1 [BD649]
      layover-airport 1.0: 1 [BD649]
        price 1.0: 1 [BD649]
          travel-time 1.0: 1 [BD649]
            arrival-time 0.0: 1 [BD649]
    number-of-legs 0.0: 3 [BA1660/BA1618, BA4039/BA4976, LH4507/
      LH4436]
      arrival-time 1.0: 3 [BA1660/BA1618, BA4039/BA4976, LH4507/
        LH4436]
        layover-airport 0.0: 3 [BA1660/BA1618, BA4039/BA4976,
          LH4507/LH4436]
          travel-time 1.0: 2 [BA1660/BA1618, BA4039/BA4976]
            price 0.0: 2 [BA1660/BA1618, BA4039/BA4976]
          travel-time 0.0: 1 [LH4507/LH4436]
            price 1.0: 1 [LH4507/LH4436]
fare-class 0.0: 2 [BD629, KL1282/KL1731]
  price 1.0: 2 [BD629, KL1282/KL1731]
    arrival-time 1.0: 2 [BD629, KL1282/KL1731]
      travel-time 1.0: 2 [BD629, KL1282/KL1731]
        number-of-legs 1.0: 1 [BD629]
          airline 0.0: 1 [BD629]
            layover-airport 1.0: 1 [BD629]
        number-of-legs 0.0: 1 [KL1282/KL1731]
          layover-airport 0.0: 1 [KL1282/KL1731]
            airline 1.0: 1 [KL1282/KL1731]
```

Appendix D

Core Data Structure

D.1 Structure Comparison FLIGHTS - my Version

FLIGHTS

```
'#S(evaluation
:prefs (
#S(pref
      :active t
      :name "fare-class"
      :ranking 1
      :weight 0.37)


#S(pref
      :active t
      ...)
...
)
:options (
#S(opt
      :name flight1
      :code "BA1660/BA1618"
      :evid #S(evidence
            :evaluation 0.72
            :z-score 1.68)


      :description (
            ( "airline-code"."KL")
            ( "airline-name"."KLM")
            ...)
      :positive (
            ( "arrival-time" .
            #S(evidence
            :evaluation 0.83
            :z-score 1.28)))
      :negative (...)


      )
#S(opt ...)
))
```

exhaustive flight property info

information relevant for compellingness

NEW VERSION

```
'#S(evaluation
:prefs (
#S(pref_new
      :active t
      :name "fare-class"
      :ranking 1
      :weight 0.37
      :prefpos ("business")
      :prefneg ())
#S(pref_new
      :active t
      ...)
...
)
:options (
#S(cluster_opt
      :name cluster1
      :code "fare-class 1.0"
      :size "4"
      :evaluation "good"
      :criterion "fare-class"
      :leaf nil
      :negsInAlt "[[arrival-time],...]"
      :reason (("fare-class" ) )
      :attributes  #S(value
        :positive (...)
        :negative (...)
        :other (
            ( "layover-airport" .
            #S(entlist :flights (
            ("BA1660/BA1618"."Manchester")
            ("BA4039/BA4976"."Bristol"))
        :further (...)
      )
      :subcluster (#S(cluster_opt
            :name cluster1.1
            ...))
      )
#S(cluster_opt ...)
))
```

additional information relevant for cluster identification

complex structure: flight lists

recursive structure

D.2 Full Data Structure after Content Selection Step

This is the data structure generated for the user model "business traveller", for the query: Flight from Brussels to Edinburgh, arriving around 6 pm on June 23rd.

```
'#S(evaluation
:prefs (
#S(pref_new  :active t :name "fare-class" :ranking 1 :weight
    0.3704081632653061 :prefpos ("business") :prefneg ())
#S(pref_new  :active t :name "number-of-legs" :ranking 2 :weight
    0.16405895691609979 :prefpos () :prefneg ())
#S(pref_new  :active t :name "arrival-time" :ranking 2 :weight
    0.16405895691609979 :prefpos ("18_00") :prefneg ())
#S(pref_new  :active t :name "travel-time" :ranking 2 :weight
    0.16405895691609979 :prefpos () :prefneg ())
#S(pref_new  :active nil :name "departure-time" :ranking 2 :weight
    0.0 :prefpos () :prefneg ())
#S(pref_new  :active t :name "airline" :ranking 6 :weight
    0.0727891156462585 :prefpos ("KLM_Airlines") :prefneg ("Ryanair")
    )
#S(pref_new  :active t :name "layover-airport" :ranking 7 :weight
    0.032312925170068035 :prefpos () :prefneg ("London_Heathrow"))
#S(pref_new  :active t :name "price" :ranking 7 :weight
    0.032312925170068035 :prefpos () :prefneg ()))
:options (
#S(cluster_opt :name cluster1 :code "fare-class␣1.0"  :size "4" :
    evaluation "good" :criterion "fare-class" :leaf nil :negsInAlt "
    [[arrival-time],␣[[layover-airport,␣[layover-airport,␣[price]]],␣
    [arrival-time]]]" :reason (  ( "fare-class") )
        :attributes  #S(value
        :positive (
          ( "fare-class" . #S(entlist :flights ( ("BD649" . "[
              business,_economy]") ("BA1660/BA1618" . "[business,
              _economy]") ("BA4039/BA4976" . "[business,_economy]") (
              "LH4507/LH4436" . "[business,_economy]"))))        )
        :negative ()
        :other (( "airline" . #S(entlist :flights ( ("BD649" . "
            BMI_British_Midland") ("BA1660/BA1618" . "British_Airways
            ") ("BA4039/BA4976" . "British_Airways") ("LH4507/LH4436"
             . "Lufthansa")))))
        :further (( "business-price" . #S(entlist :flights ( ("BD649
            " . "256,20GBP") ("BA1660/BA1618" . "308,90GBP") ("BA4039
            /BA4976" . "308,90GBP") ("LH4507/LH4436" . "270,30GBP")))
            )
        ( "number-of-legs" . #S(entlist :flights ( ("BD649" . "1") (
            "BA1660/BA1618" . "2") ("BA4039/BA4976" . "2") ("LH4507/
            LH4436" . "2"))))
        ( "layover-airport" . #S(entlist :flights ( ("BD649" . "none
            ") ("BA1660/BA1618" . "Manchester") ("BA4039/BA4976" . "
            Bristol") ("LH4507/LH4436" . "Frankfurt"))))
        ( "arrival-time" . #S(entlist :flights ( ("BD649" . "20_30")
             ("BA1660/BA1618" . "16_20") ("BA4039/BA4976" . "18_30")
            ("LH4507/LH4436" . "18_10"))))
        ( "travel-time" . #S(entlist :flights ( ("BD649" . "01_35")
```

```
        ("BA1660/BA1618" . "03_20") ("BA4039/BA4976" . "03_30") (
        "LH4507/LH4436" . "04_35"))))
    ( "economy-price" . #S(entlist :flights ( ("BD649" . "111,70
        GBP") ("BA1660/BA1618" . "221,90GBP") ("BA4039/BA4976" .
        "221,90GBP") ("LH4507/LH4436" . "114,30GBP"))))
    ( "departure-time" . #S(entlist :flights ( ("BD649" . "17_55
        ") ("BA1660/BA1618" . "12_00") ("BA4039/BA4976" . "14_00"
        ) ("LH4507/LH4436" . "12_35"))))
    ) ) :subcluster (
#S(cluster_opt :name cluster1.1 :code "number-of-legs␣1.0" :size "1
  " :evaluation "good" :criterion "number-of-legs" :leaf nil :
  negsInAlt "[arrival-time]" :reason ( ( "number-of-legs") )
    :attributes #S(value
    :positive (
      ( "number-of-legs" . #S(entlist :flights ( ("BD649" . "1")
          )))
    ( "layover-airport" . #S(entlist :flights ( ("BD649" . "none
        ")))))
    :negative ()
    :other (( "arrival-time" . #S(entlist :flights ( ("BD649" .
        "20_30")))))
    :further (( "business-price" . #S(entlist :flights ( ("BD649
        " . "256,20GBP"))))
    ( "fare-class" . #S(entlist :flights ( ("BD649" . "[business
        ,_economy]"))))
    ( "airline" . #S(entlist :flights ( ("BD649" . "
        BMI_British_Midland"))))
    ( "travel-time" . #S(entlist :flights ( ("BD649" . "01_35"))
        ))
    ( "economy-price" . #S(entlist :flights ( ("BD649" . "111,70
        GBP"))))
    ( "departure-time" . #S(entlist :flights ( ("BD649" . "17_55
        "))))))
    :subcluster ())
#S(cluster_opt :name cluster1.2 :code "number-of-legs␣0.0" :size "3
  " :evaluation "average" :criterion "number-of-legs" :leaf nil :
  negsInAlt "[[layover-airport,␣[layover-airport,␣[price]]],␣[
  arrival-time]]" :reason ( ( "arrival-time") )
    :attributes #S(value
    :positive (( "arrival-time" . #S(entlist :flights ( ("BA1660
        /BA1618" . "16_20") ("BA4039/BA4976" . "18_30") ("LH4507/
        LH4436" . "18_10")))))
    :negative ()
    :other (( "number-of-legs" . #S(entlist :flights ( ("BA1660/
        BA1618" . "2") ("BA4039/BA4976" . "2") ("LH4507/LH4436" .
        "2"))))
    ( "layover-airport" . #S(entlist :flights ( ("BA1660/BA1618"
        . "Manchester") ("BA4039/BA4976" . "Bristol") ("LH4507/
        LH4436" . "Frankfurt")))))
    :further (( "business-price" . #S(entlist :flights ( ("
        BA1660/BA1618" . "308,90GBP") ("BA4039/BA4976" . "308,90
        GBP") ("LH4507/LH4436" . "270,30GBP"))))
    ( "fare-class" . #S(entlist :flights ( ("BA1660/BA1618" . "[
        business,_economy]") ("BA4039/BA4976" . "[business,
        _economy]") ("LH4507/LH4436" . "[business,_economy]"))))
    ( "airline" . #S(entlist :flights ( ("BA1660/BA1618" . "
```

```
            British_Airways") ("BA4039/BA4976" . "British_Airways") (
            "LH4507/LH4436" . "Lufthansa"))))
        ( "travel-time" . #S(entlist :flights ( ("BA1660/BA1618" . "
            03_20") ("BA4039/BA4976" . "03_30") ("LH4507/LH4436" . "
            04_35"))))
        ( "economy-price" . #S(entlist :flights ( ("BA1660/BA1618" .
             "221,90GBP") ("BA4039/BA4976" . "221,90GBP") ("LH4507/
            LH4436" . "114,30GBP"))))
        ( "departure-time" . #S(entlist :flights ( ("BA1660/BA1618"
            . "12_00") ("BA4039/BA4976" . "14_00") ("LH4507/LH4436" .
             "12_35"))))
        ) ) :subcluster ()))))
#S(cluster_opt :name cluster2 :code "fare-class_0.0"  :size "2" :
    evaluation "average" :criterion "fare-class" :leaf nil :negsInAlt
     "[[arrival-time,_[travel-time,_[airline]]]]" :reason (  (  "
    number-of-legs" "arrival-time" "layover-airport") (  "airline") )
        :attributes  #S(value
        :positive (( "arrival-time" . #S(entlist :flights ( ("BD629"
             . "16_10") ("KL1282/KL1731" . "16_50")))))
        :negative ()
        :other (( "fare-class" . #S(entlist :flights ( ("BD629" . "
            economy") ("KL1282/KL1731" . "economy")))))
        :further (( "number-of-legs" . #S(entlist :flights ( ("BD629
            " . "1") ("KL1282/KL1731" . "2"))))
        ( "layover-airport" . #S(entlist :flights ( ("BD629" . "none
            ") ("KL1282/KL1731" . "Amsterdam"))))
        ( "airline" . #S(entlist :flights ( ("BD629" . "
            BMI_British_Midland") ("KL1282/KL1731" . "KLM_Airlines"))
            ))
        ( "travel-time" . #S(entlist :flights ( ("BD629" . "01_35")
            ("KL1282/KL1731" . "03_10"))))
        ( "economy-price" . #S(entlist :flights ( ("BD629" . "111,70
            GBP") ("KL1282/KL1731" . "107,70GBP"))))
        ( "departure-time" . #S(entlist :flights ( ("BD629" . "13_35
            ") ("KL1282/KL1731" . "12_40"))))
        ) ) :subcluster (
#S(cluster_opt :name cluster2.1 :code "number-of-legs_1.0"  :size "1
    " :evaluation "good" :criterion "number-of-legs" :leaf nil :
    negsInAlt "[]" :reason (("number-of-legs" "layover-airport"))
        :attributes  #S(value
        :positive (( "number-of-legs" . #S(entlist :flights ( ("
            BD629" . "1"))))
        ( "layover-airport" . #S(entlist :flights ( ("BD629" . "none
            ")))))
        :negative ()
        :other (( "airline" . #S(entlist :flights ( ("BD629" . "
            BMI_British_Midland")))))
        :further (( "fare-class" . #S(entlist :flights ( ("BD629" .
            "economy"))))
        ( "arrival-time" . #S(entlist :flights ( ("BD629" . "16_10")
            )))
        ( "travel-time" . #S(entlist :flights ( ("BD629" . "01_35"))
            ))
        ( "economy-price" . #S(entlist :flights ( ("BD629" . "111,70
            GBP"))))
        ( "departure-time" . #S(entlist :flights ( ("BD629" . "13_35
```

```
            "))))
        ) ) :subcluster ())
#S(cluster_opt :name cluster2.2 :code "number-of-legs␣0.0"  :size "1
    " :evaluation "average" :criterion "number-of-legs" :leaf nil :
    negsInAlt "[arrival-time,␣[travel-time,␣[airline]]]" :reason (  (
      "airline") )
        :attributes  #S(value
        :positive (( "airline" . #S(entlist :flights ( ("KL1282/
            KL1731" . "KLM_Airlines")))))
        :negative ()
        :other (( "number-of-legs" . #S(entlist :flights ( ("KL1282/
            KL1731" . "2"))))
        ( "layover-airport" . #S(entlist :flights ( ("KL1282/KL1731"
             . "Amsterdam")))))
        :further (( "fare-class" . #S(entlist :flights ( ("KL1282/
            KL1731" . "economy"))))
        ( "arrival-time" . #S(entlist :flights ( ("KL1282/KL1731" .
            "16_50"))))
        ( "travel-time" . #S(entlist :flights ( ("KL1282/KL1731" . "
            03_10"))))
        ( "economy-price" . #S(entlist :flights ( ("KL1282/KL1731" .
             "107,70GBP"))))
        ( "departure-time" . #S(entlist :flights ( ("KL1282/KL1731"
            . "12_40"))))
        ) ) :subcluster ()))))
```

Appendix E

Content Plan

Plan start begins
Plan start ends
(describe_clusters list_flights) begins
(turn list_flights) begins
(sequence list_flights) begins
Dummy node-3-1-1-1 begins
(elaborate cluster1) begins
(suggest cluster1) begins
(identify cluster1) begins
(inform cluster cluster1 size_4 status_given definite_false info_rheme) begins
(inform cluster cluster1 size_4 status_given definite_false info_rheme) ends
(inform (business_class available) cluster1) begins
(inform (business_class available) cluster1) ends
(identify cluster1) ends
(suggest cluster1) ends
(list_other cluster1) begins
Dummy node-3-1-1-2-2-1 begins
(inform (airline (not KLM_Airlines)) cluster1) begins
(inform (airline (not KLM_Airlines)) cluster1) ends
Dummy node-3-1-1-2-2-1 ends
(list_other cluster1) ends
(elaborate cluster1) ends
(elaborate cluster1.1) begins
(suggest cluster1.1) begins
(identify cluster1.1) begins
(inform cluster cluster1.1 size_1 status_given definite_false info_rheme) begins
(inform cluster cluster1.1 size_1 status_given definite_false info_rheme) ends
(inform (direct) cluster1.1) begins
(inform (direct) cluster1.1) ends
(identify cluster1.1) ends
(suggest cluster1.1) ends
(list_other cluster1.1) begins
Dummy node-3-1-1-3-2-1 begins
(inform (arrival-time 20_30) cluster1.1) begins
(inform (arrival-time 20_30) cluster1.1) ends
Dummy node-3-1-1-3-2-1 ends
(list_other cluster1.1) ends

(elaborate cluster1.1) ends
(elaborate cluster1.2) begins
(suggest cluster1.2) begins
(identify cluster1.2) begins
(inform cluster cluster1.2 size_3 status_given definite_false info_theme) begins
(inform cluster cluster1.2 size_3 status_given definite_false info_theme) ends
(inform (arrival-time eval=earlier) cluster1.2) begins
(inform (arrival-time eval=earlier) cluster1.2) ends
(identify cluster1.2) ends
(suggest cluster1.2) ends
(list_other cluster1.2) begins
Dummy node-3-1-1-4-2-1 begins
(inform (indirect) cluster1.2) begins
(inform (indirect) cluster1.2) ends
(inform (connection (not London_Heathrow)) cluster1.2) begins
(inform (connection (not London_Heathrow)) cluster1.2) ends
Dummy node-3-1-1-4-2-1 ends
(list_other cluster1.2) ends
(elaborate cluster1.2) ends
(elaborate cluster2) begins
(suggest cluster2) begins
(identify cluster2) begins
(inform cluster cluster2 size_2 status_given definite_false info_rheme val_neg) begins
(inform cluster cluster2 size_2 status_given definite_false info_rheme val_neg) ends
(inform (economy_class available) cluster2) begins
(inform (economy_class available) cluster2) ends
(identify cluster2) ends
(suggest cluster2) ends
(list_comp cluster2) begins
Dummy node-3-1-1-5-2-1 begins
(inform (arrival-time eval=good) cluster2) begins
(inform (arrival-time eval=good) cluster2) ends
Dummy node-3-1-1-5-2-1 ends
(list_comp cluster2) ends
(elaborate cluster2) ends
(elaborate cluster2.1) begins
(suggest cluster2.1) begins
(identify cluster2.1) begins
(inform cluster cluster2.1 size_1 status_given definite_false info_rheme) begins
(inform cluster cluster2.1 size_1 status_given definite_false info_rheme) ends
(inform (direct) cluster2.1) begins
(inform (direct) cluster2.1) ends
(identify cluster2.1) ends
(suggest cluster2.1) ends
(list_other cluster2.1) begins
Dummy node-3-1-1-6-2-1 begins
(inform (airline BMI_British_Midland) cluster2.1) begins
(inform (airline BMI_British_Midland) cluster2.1) ends
Dummy node-3-1-1-6-2-1 ends
(list_other cluster2.1) ends
(elaborate cluster2.1) ends
(elaborate cluster2.2) begins
(suggest cluster2.2) begins

(identify cluster2.2) begins
(inform cluster cluster2.2 size_1 status_given definite_false info_theme) begins
(inform cluster cluster2.2 size_1 status_given definite_false info_theme) ends
(inform (airline KLM_Airlines) cluster2.2) begins
(inform (airline KLM_Airlines) cluster2.2) ends
(identify cluster2.2) ends
(suggest cluster2.2) ends
(list_other cluster2.2) begins
Dummy node-3-1-1-7-2-1 begins
(inform (connection Amsterdam) cluster2.2) begins
(inform (connection Amsterdam) cluster2.2) ends
Dummy node-3-1-1-7-2-1 ends
(list_other cluster2.2) ends
(elaborate cluster2.2) ends
Dummy node-3-1-1-1 ends
(sequence list_flights) ends
(turn list_flights) ends
(release_turn) begins
(release_turn) ends
(no_wait) begins
(no_wait) ends
(describe_clusters list_flights) ends
Plan finish begins
Plan finish ends

Appendix F

Sentence Plan

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Here are the LFs for each sentence. -->
<turn>

 <!-- I found 4_H* flights with availability in
     business_class_H* LL% , but none_H* are on KLM_H* LL% . :-
      sent
 from content plan tree branch:
  <suggest arg="cluster1">
    <identify arg="cluster1">
      <inform arg="cluster1" definite="false" info="rheme"
         pred="cluster" size="4" status="given"/>
      <inform arg="cluster1" pred="business-class" val="
         available"/>
    </identify>
  </suggest>
  <concede arg="cluster1">
    <inform arg="cluster1" negation="true" pred="airline" val=
       "KLM"/>
  </concede> --->

  <sent>
    <lf>
      <node info="rh" mood="dcl" id="n1" pred="but">
        <rel name="Arg1">
          <one-of>
            <node id="n2" pred="find" tense="past">
              <rel name="Perceiver">
                <node id="n3" pred="pro1" num="sg"/>
              </rel>
              <rel name="Sought">
                <node id="n4" pred="flight" num="pl" det="nil"
                   >
```

```
            <rel name="Card">
              <node id="n5" pred="4" kon="+"/>
            </rel>
            <rel name="Poss">
              <node id="n6" pred="availability" det="nil
                  ">
                <rel name="Category">
                  <node id="n7" pred="business_class"
                      kon="+"/>
                </rel>
              </node>
            </rel>
          </node>
        </rel>
      </node>
      <node id="n8" pred="there-be" info="rh" mood="dcl"
          tense="pres">
        <rel name="Arg">
          <node id="n9" pred="flight" num="pl" det="nil"
              >
            <rel name="Card">
              <node id="n10" pred="4" kon="+"/>
            </rel>
            <rel name="Poss">
              <node id="n11" pred="availability" det="
                  nil">
                <rel name="Category">
                  <node id="n12" pred="business_class"
                      kon="+"/>
                </rel>
              </node>
            </rel>
          </node>
        </rel>
      </node>
    </one-of>
  </rel>
  <rel name="Arg2">
    <node id="n14" pred="none" kon="+">
      <rel name="Restr">
        <node idref="n13"/>
      </rel>
      <rel name="Body">
        <node id="n15" pred="be" tense="pres">
          <rel name="Arg">
            <node idref="n13"/>
          </rel>
          <rel name="Prop">
            <node id="n16" pred="has-rel">
```

```
                    <rel name="Of">
                      <node idref="n13"/>
                    </rel>
                    <rel name="Airline">
                      <node id="n17" pred="KLM" kon="+"/>
                    </rel>
                  </node>
                </rel>
              </node>
            </rel>
          </node>
        </rel>
      </node>
    </lf>
  </sent>
```

Appendix G

Evaluation Dialogue Pairs

G.1 Dialogue 1 (Carol 1)

Carol's user profile2:

You're going to read the transcripts of dialogues between a customer, Carol, and two different automated travel agents. When asked to fill out a form indicating his preferences for air travel, here's what Carol said:
I'm a frequent flier and collect frequent flier points with KLM, so the most important thing when booking a flight is the airline. Second most important are the arrival time and the number of legs. Third most important is the travel time. I'm not too concerned about the price or the layover airport, although I prefer not to connect in London Heathrow.

Dialogue between Carol and Demberg system:

1User:
Hello. I would like to book a flight from Edinburgh to San Francisco for Thursday, 27th July, preferably leaving Edinburgh after 11 am. If at all possible, I'd like to fly on KLM.

1System:
Do you have any preferences for airports?

2User:
I'd rather not connect in London Heathrow.

2System:
I found two flights that both require a connection. There' s a KLM flight for only 348 pounds with a connection in Amsterdam, but it takes 18 hours and 20 minutes and leaves already at 7:30am.

To leave later and travel faster, you can take the Flybe flight that leaves only at 11:45 am and has a total travel-time of only 15hours 40 minutes. But you'd need to connect in London and the price is 580 pounds.

Dialogue between Carol and Polifroni system

1User:
Hello. I would like to book a flight from Edinburgh to San Francisco for Thursday, 27th July, preferably leaving Edinburgh after 11 am. If at all possible, I'd like to fly on KLM.

1System:
There are no flights that leave Edinburgh after 11 am and that are on KLM. However, there is a KLM flight leaving at 7:30am.

2User:
Are there any flights on other airlines that leave Edinburgh after 11 am?

2System: There are 7 flights that leave Edinburgh after 11 am. All of them have availability in business class.

3User:
Do you have information about the travel time of these flights?

3System: Most of the flights take between 18 hours and 20 hours, one has a travel time of less than 16 hours.

4User:
Please tell me about that flight with under 16 hours travel time.

4System:
The Flybe flight leaves Edinburgh at 11:45 am and arrives San Francisco at 11 pm. It has a total travel-time of 15hours 40 minutes and costs 580 pounds. You'd need to connect in London Heathrow.

G.2 Dialogue 2 (Walter1)

Walter's user profile:

You're going to read the transcripts of dialogues between a customer, Walter, and two different automated travel agents. When asked to fill out a form indicating his preferences for air travel, here's what Walter said:
I'm a business traveller, so the most important thing for me is to fly business class. I

need to get to my meetings in time and don't want to spend too much time on airports, so second most important to me are arrival time, travel time and number of legs. Third most important is to fly on KLM, because I collect bonus miles. I'm not concerned about travelling cheaply.

Dialogue between Walter and Demberg system:

1Walter:
Hello. I'd like to book a flight from Edinburgh to Brussels for tomorrow. I need to get there at around 3pm and I'd prefer to fly business class.

1System:
Do you have any airline preferences?

2Walter:
Yes, KLM. And I'd rather not connect in London, if at all possible.

2System:
There are four flights with availability in business class. None are on KLM. The only direct flight arrives at 5:30 pm, which is later than you requested. To arrive earlier, you'll have to make a connection.
If you're willing to travel economy, there is a direct flight on BMI, arriving at 1:10 pm. There's also a KLM flight arriving at 1:50 pm, but it requires a connection in Amsterdam.
All other economy flights take longer or arrive later than 4 pm.

3Walter:
Can you tell me more about the business class flights that arrive earlier?

3System:
The two flights with shortest travel time are both on British Airways and cost 308 pounds. The first takes 4 hours and 20 minutes and arrives at 1:20 pm. The other takes 4 hours and 30 minutes and arrives at 3:30pm.
There's a cheaper flight on Lufthansa. It arrives at 3:10pm and costs just 270 pounds, but it takes 5 hours and 35 minutes.
All other business class flights arrive much too early or very late.

Dialogue between Walter and Polifroni System:

1Walter:
I'd like to book a flight from Edinburgh to Brussels for tomorrow.

1System:

There are 28 flights from Edinburgh to Brussels. Most of them require a connection and cost between 100 and 230 pounds.

2Walter:
Are there any direct flights with availability in business class that arrive around 3 pm?

2System:
There are no direct business class flights that arrive around 3 pm. However, there is one direct business class flight that arrives at 5:30. It costs 111 pounds.

3Walter:
No, that's too late... what about connecting business class flights that arrive around 3 pm?

3System:
There are three business class flights that require a connection and arrive around 3 pm. Two of them are on British Airways and arrive at 1:20 and 3:30. The third is on Lufthansa and arrives at 3:10.

4Walter:
How long do these flights take?

4System:
The British Airways flights take 3 hours and 20 minutes and the Lufthansa flight takes 4 hours and 35 minutes.

5Walter:
Are there any direct economy flights?

5System:
Yes, there's a direct flight on BMI. It arrives at 1:10pm and has a travel time of 1 hour 35 minutes.

G.3 Dialogue 3 (Carol 2)

Carol's user profile2:

It's Carol again. This time she wants information about flights to Brussels. Remember what she said in her user profile:I'm a frequent flier, so the first most important thing for me is the airline, because I collect frequent flier points with KLM. Second most important to me is arrival time and number of legs. Third most important is travel-time. I'm not too concerned about the price or the layover airport, although I prefer not to connect in London Heathrow.

Dialogue between Carol and Demberg system:

1User:
Hello. I would like to book a flight from Edinburgh to Brussels for Thursday, 27th July, arriving at about noon. I prefer KLM flights.

1System:
I found three flights with a good price. Two of them are on KLM. They both cost 111 pounds, but require a connection in Amsterdam. If you want to fly direct, there's a BMI flight arriving Brussels at 9:40 am that costs 107 pounds. All other flights arrive after noon.

2User:
Please tell me more about the KLM flights.

2System:
One of them arrives at 11:20 am but has a travel-time of 4hours 15 minutes. The other takes just 3hours 10 minutes but arrives only at 4:40 pm.

Dialogue between Carol and Polifroni system

1User:
Hello. I would like to book a flight from Edinburgh to Brussels for Thursday, 27th July, arriving at about noon. I prefer KLM flights.

1System:
I found one KLM flight that arrives Brussels at 11:20 am. It takes 4h 15 minutes.

2User:
Please tell me more about that flight.

2System:
It leaves Edinburgh at 6:05 am and costs 111 Pounds. You need to make a connection in Amsterdam.

G.4 Dialogue 4 (Steve 2)

Steve's user profile2:

It's Steve again. This time he wants to go to San Francisco. Remember Steve's user profile:
I'm a student, so the most important thing to me is to get a cheap flight. Otherwise I'm pretty flexible. I have no preferred airline and I don't care about particular layover

airports.

Dialogue between Steve and Demberg system: 1User:
Hello. I would like to book a flight from Edinburgh to Brussels for Thursday, 27th July. Please find me a cheap flight.

1System:
What is your desired arrival time?

2User:
Ahm... in the early afternoon.

2System:
Ryan Air offers the cheapest flights to Brussels. They cost just 49 pounds, but you would have to connect in Dublin. There are two flights to choose from. There's an early flight, leaving Edinburgh at 8:05am and arriving at 12:45pm, with a travel time of only 3 hours 30 minutes. If you prefer to arrive at 5:30pm, your total travel time would be 5 hours 30 minutes, leaving Edinburgh at 1pm.
If you want to fly direct, there's a flight on BMI that takes just 1 hour and 35 minutes. It arrives at 1:35pm and costs 112 pounds. All other direct flights are more expensive.

Dialogue between Steve and Polifroni System:

1User:
Hello. I would like to book a flight from Edinburgh to Brussels for Thursday, 27th July. Please find me a cheap flight.

System:
There are three flights for only 49 pounds on Ryan Air. They all require a connection in Dublin.

User:
Is there also a cheap *direct* flight?

System:
The cheapest direct flight is on BMI and costs 112 pounds.

User:
Okay... what are the departure and arrival times of the three RyanAir flights?

System:
The first Ryan Air leaves Edinburgh at 8:05 am and arrives at 12:45 pm. The next one leaves at 1 pm and arrives at 5:30pm. The last one leaves Edinburgh at 4:25 pm and arrives at 9:25 pm.

G.5 Dialogue 5 (Walter 2)

Walter's User Profile

It's [UName] again. This time he has another request. Remember his preferences:
I'm a business traveller, so the most important thing for me is to fly business class. I need to get to my meetings in time and don't want to spend too much time on airports, so second most important to me are arrival time, travel time and number of legs. Third most important is to fly on KLM, because I collect bonus miles. I'm not concerned about travelling cheaply.

Dialogue between Walter and Demberg system

1User:
Hello. I would like to book a flight from Edinburgh to San Francisco for tomorrow, arriving around 4 pm. It's important for me to fly on business class. A KLM flight would be nice.

1System:
There are two flights that arrive San Francisco at 3:45 and 4:35 but neither of them is direct. They both have availability in business class and take under 15 hours. One of them is on Continental Airlines and the other is on Air France.
There's also a KLM flight, but its travel time is 18 h 20 min.
All other flights arrive later than you requested.

2User:
Please tell me more about the Air France flight.

2System:
The Air France flight leaves Edinburgh at 9:05 am and arrives San Francisco at 3:45 pm. You need to connect in Paris. The flight costs 1.611 pounds on business class.

Dialogue between Walter and Polifroni system

1Walter:
I'd like to book a flight from Edinburgh to San Francisco for tomorrow, arriving around 4 pm. It's important for me to fly on business class. A KLM flight would be nice.

1System:
I found one KLM flight with availability in business class. It arrives at 3:45. You'd need to connect in Amsterdam.

2Walter:
Are there any direct flights that arrive around 4 pm?

2System:
There are no direct flights from Edinburgh to San Francisco. However, there are 7 flights that arrive San Francisco between 3 and 5 pm. They all have availability on business class.

3Walter:
How long does the KLM flight take?

3System: The KLM flight has a travel time of 18hours 20.

4Walter:
18 h is very long. Are any of those other 7 flights faster?

4System:
There are three flights that take under 15 hours. The British Airways flight arrives at 4:10 pm, the Continental Airways flight arrives at 4:35 pm and the Air France flight arrives at 3:45 pm.

5Walter:
Please tell me more about the Air France Flight.

5System:
The Air France flight leaves Edinburgh at 9:05 am and arrives San Francisco at 3:45 pm. You need to connect in Paris. The flight costs 1.611 pounds on business class.

Bibliography

Artificial Intelligence Applications Institute, University of Edinburgh. *O-Plan Task Formalism Manual*, 1997.

Artificial Intelligence Applications Institute, University of Edinburgh. *O-Plan User Guide*, 1997.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. Dipper: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, 2003.

Cem Bozsahin, Geert-Jan M. Kruijff, and Michael White. Specifying grammars for openccg. March 2005.

Giuseppe Carenini and Johanna D. Moore. An empirical study of the influence of user tailoring on evaluative argument effectiveness. In *Proceedings of IJCAI 2001, Seattle, USA*, 2001.

Giuseppe Carenini and Johanna D. Moore. Generating and evaluating evaluative arguments. submitted, 2004.

Giuseppe Carenini. Evaluating user-tailored evaluative arguments: Persuasive effectiveness and its communicative appropriateness. In *Proceedings of the IJCAI Workshop on Computational Models of Natural Argument, Acapulco, Mexico*, 2003.

Christian Chiarcos and Manfred Stede. Salience-driven text planning. In *International Conference on Natural Language Generation*, pages 21 – 30, 2004.

Grace Chung. Developing a flexible spoken dialog system using simulation. In *Proceedings of ACL '04, Barcelona, Spain*, July 2004.

Ken Currie and Austin Tate. O-plan the open planning architecture. *Artificial Intelligence*, 52:49–86, 1991.

Mary Ellen Foster and Michael White. Techniques for text planning with xslt. In *Proceedings, Fourth NLPXML Workshop, Barcelona*, July 2004.

Anil K. Jain, Narasimha M. Murty, and Patrick J. Flynn. *Data Clustering: A Review*, volume 31, pages 264–323. The Ohio State University, September 1999.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, chapter 14 Clustering, pages 500 – 528. The MIT Press, 1999.

David L. Martin, Adam J. Cheyer, and Douglas B. Moran. Bilding distributed software systems with the open agent architecture. In *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1998.

Johanna D. Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. Generating tailored, comparative descriptions in spoken dialogue. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Sociey Conference, AAAI Press*, 2004.

Joseph Polifroni, Grace Chung, and Stephanie Seneff. Towards automatic generation of mixed-initiative dialogue systems from web content. In *Proceedings of Eurospeech '03, Geneva, Switzerland*, pages 193 – 196, September 2003.

Scott Prevost and Mark Steedman. Specifying intonation from context for speech synthesis. In *Speech Communication*, pages 139–153, 1994.

Scott Prevost. An information structural approach to spoken language generation. In *Proceedings of the 34th ACL, Santa Cruz*, pages 294–301, 1996.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. In *Journal of Natural Language Engineering*, volume 3, pages 57–87, 1997.

Walter Schneider, Amy Eschman, and Anthony Zuccolotto. *E-Prime User's Guide*. Pittsburgh: Psychology Software Tools, Inc., 2002.

Amanda Stent, Marilyn Walker, Steve Whittaker, and Preetam Maloor. User-tailored generation for spoken dialogue: an experiment. In *Proceedings of ICSLP-02*, 2002.

Marilyn A. Walker, Rebecca Passonneau, and Julie E. Boland. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of ACL-01*, 2001.

Marilyn A. Walker, Steve Whittaker, Amanda Stent, Preetam Maloor, Johanna D. Moore, Michael Johnston, and Gunaranjan Vasireddy. Speech plans: generating evaluative responses in spoken dialogue. In *International Natural Language Generation Conference*, pages 73 – 80, 2002.

Marilyn A. Walker, Steve Whittaker, Amanda Stent, Preetam Maloor, Johanna D. Moore, Michael Johnston, and Gunaranjan Vasireddy. Generation and evaluation of user tailored responses in dialogue. *Cognitive Science*, 2004.